REVISTA PARA USUARIOS DE

Drean Commodore

AÑO 1 Nº 7 ₳ 2.00 REP. ARGENTINA



JUNIO 1986 Nº 15 ₳ 2,30 REP. ARGENTINA



COMPUTACION PARA TODOS

Número Aniversario

Regalo:

Poster Gigante de Silicon Valley

K-64 en Silicon Valley

Producción Nacional de Hardware y de Software

Guía de Computadoras, Accesorios y Servicios para Commodore, TS, CZ, TK, MSX y TI.

Todo Sobre Bases de Datos

¿Quién Ganará el Lingote de Oro?

SUMARIO

NOTAS TECNICAS



En el número anterior hemos explicado algunas formas de ordenar información. En éste comentamos dos métodos para poder hallar datos.

PROGRAMAS

Las máquinas "inteligentes" están siendo actualmente utilizadas para efectuar diagnósticos y resolver problemas tanto en la medicina como en la industria. Incluso algunos de los productores de software están haciendo experiencias en ese campo.

REVISION DE SOFT



Electronic Arts

El truco llega a la computación y nosotros se los presentamos. Además peleamos kung-fú y como si fuera poco volamos en jet como un "lobo del aire".

SECCIONES FIJAS

Drean commodore AÑO 1 Nº 7 JUNIO DE 1986

Director General
Ernesto del Castillo

Director Editorial
Cristián Pusso

Director Periodístico
Fernando Flores

Director Financiero:
Javier Campos Malbrán

Arte y Diagramación
Mario de Mendoza
Tamara Migelson

Fotografía
Víctor Grubicy

Coordinador
Ariel Testori

Redacción
Cristián Parodi

Departamento de Avisos
Oscar Devoto

Departamento de Publicidad
Guillermo González Aldalur

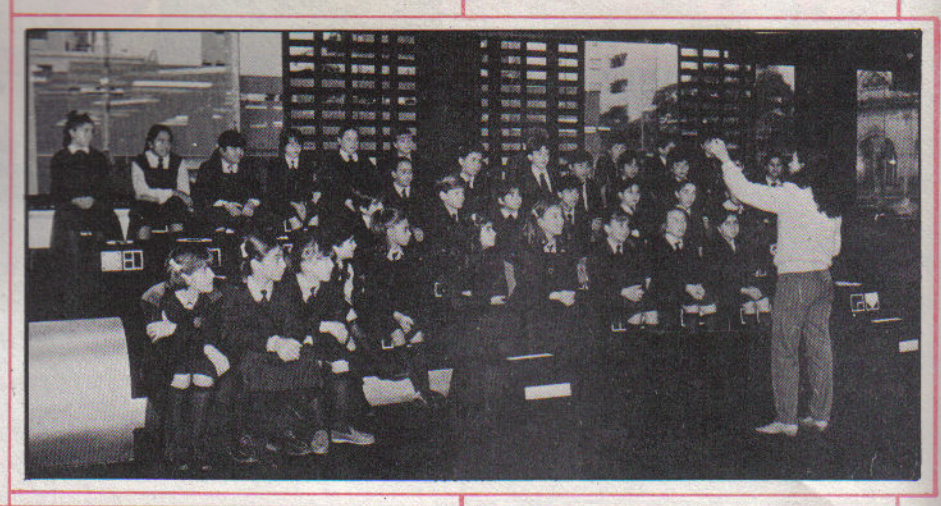
Drean Commodore es una Revista mensual editada por editorial PROEDI S.A., Paraná 720, 5º Pis. (1017) Buenos Aires. Tel.: 46-2886 y 49-7130. Reg. Nac. de la Prop. Intelectual E.T., M. Registrada.

Precio de este ejemplar. ₳ 2.
Impresión: Calcotam. Fotocromo tapa: Columbia. Fotocomposición: Van Waveren.
Los ejemplares atrasados se venderán al precio del último número en circulación.

Distribuidor en Capital: Martino, Juan de Garay 358, P.B. Capital. Distribuidor interior: DGP, Hipólito Yrigoyen 1450, Capital Federal, T.E. 38-9266/9800.

Cursos de computación

A través de un convenio con la Empresa Drean, SCIOLI ha instalado un moderno centro de enseñanza de computación usando los equipos Drean Commodore para tal fin.
El curso, denominado "Introducción al mundo de la computación", está orientado para alumnos primarios y secundarios y para los hijos de los socios.
Las clases son teórico-prácticas y, además, totalmente gratuitas. El sistema permite enseñarle, mensualmente, a 1400 alumnos. De acuerdo a las autoridades de Scioli, ya están cubiertas las vacantes para todo el año '86.



La C-64 administra supermercados

El dueño de un pequeño supermercado en los Estados Unidos, desarrolló un sistema de administración de su local utilizando una C-64. Esta trabaja con una expansión de memoria de 1 Mb. El sistema se encarga básicamente de atender los requerimientos de las dos cajas registradoras, como ser aceptación de importes de la mercadería, control de stock, totales parciales, etc. El origen del funcionamiento es a base de FIFO (First In-First Out; primero en entrar, primero en salir). Como ven, cada vez son mayores las aplicaciones de la C-64.

Lenguajes

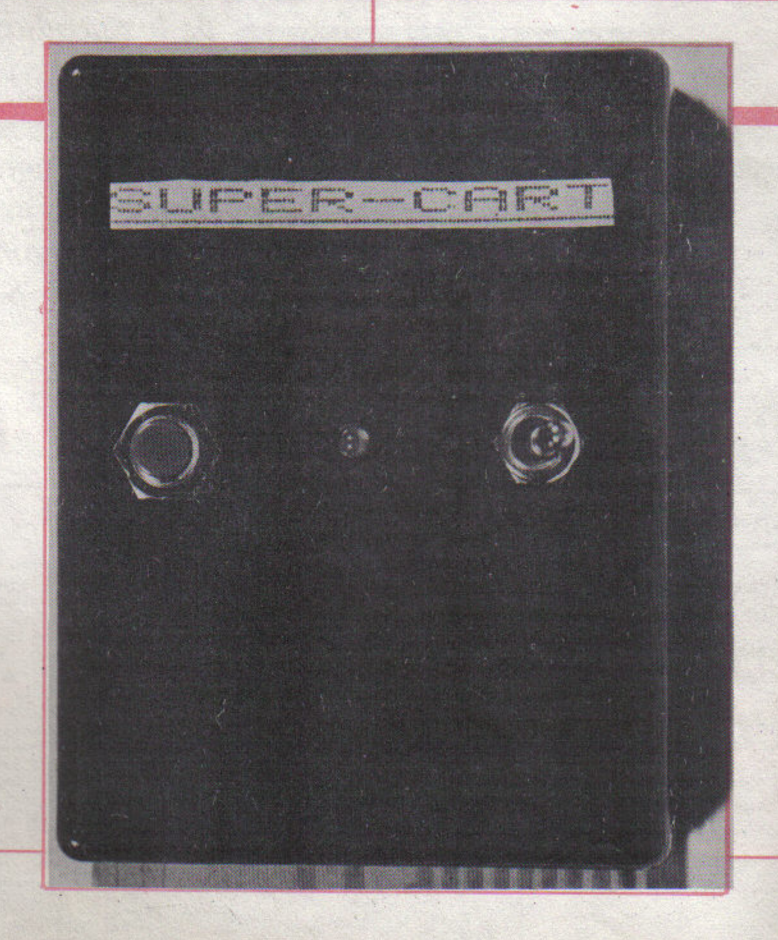
Abacus Software, la famosa empresa norteamericana de soft para los equipos Commodore, sigue lanzando nuevos productos.
En esta ocasión se trata de los lenguajes Pascal y C los cuales fueron identificados con Super Pascal y Super C, ambos en su versión para la C-128.
Como recordarán nuestros lectores, estos lenguajes ya han sido lanzados para la C-64. Ahora le toca el turno a la C-128.

Además de estos dos lenguajes, Abacus adaptó su nuevo compilador Basic también para la C-128 (existe una versión para la C-64).

De acuerdo a las especificaciones técnicas de este compilador, los programas pueden ser ejecutados de 5 a 35 veces más rápido que lo normal, es decir usando el intérprete de la C-128. Además, comenzó a publicar una serie de libros técnicos (todos para la C-128), alguno de los cuales han llegado a nuestro país.

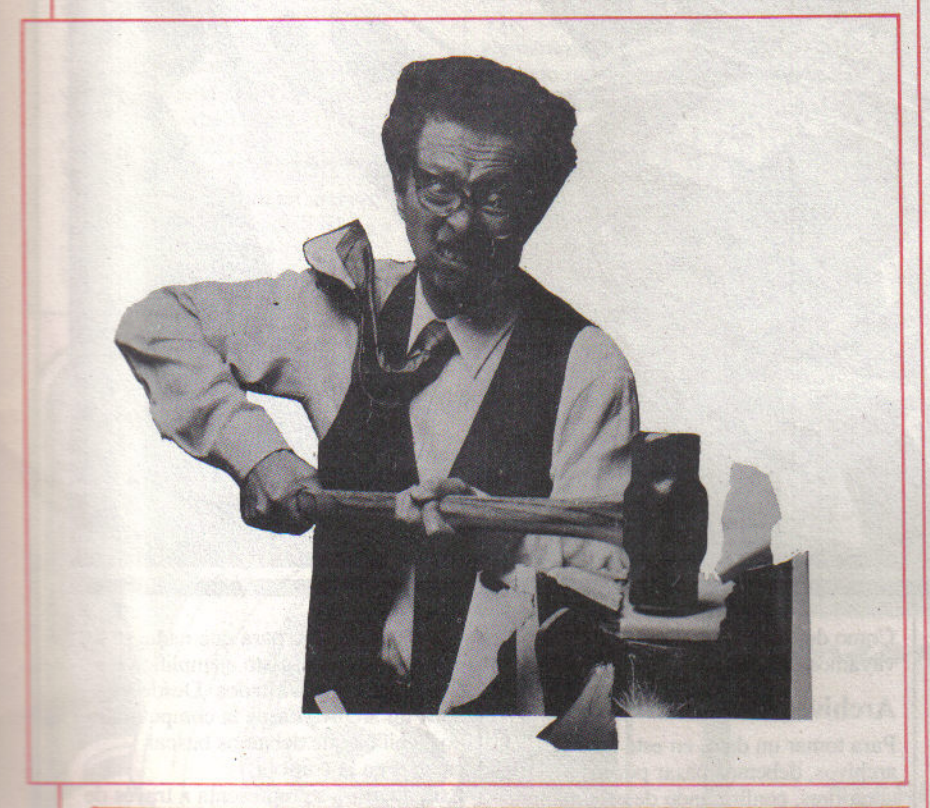
Super cart

Nuestra publicación ve con sumo interés el desarrollo de productos de software y hardware para la Drean Commodore 64.
Ese interés aumenta cuando ese producto es diseñado y desarrollado por argentinos.
S.C.e (firma Argentina) ha desarrollado un cartdridge para la Drean Commodore 64 el cual reúne en su interior una serie de programas utilitarios que S.C.e. puso dentro de una memoria ROM.
Esto es, como lo dice su manual, de dominio público. Algunos de ellos son Fast Disk, Turbo Tape y Turboplus.
También dispone de comandos que permiten recuperar programas previamente borrados a través del comando NEW, repetición de teclas, conversión de números de una base a otra, etc.
El programa ocupa un total de 8 Kb.
Lo acompaña un manual de 7 hojas.



SALVANDO PROGRAMAS

¿Quién no ha tipeado, alguna vez, el comando NEW luego de ingresar 1000 líneas de programa en la memoria? Lean esta nota y sabrán cómo evitar el suicidio.



En mayor o menor grado todos hemos hecho un NEW creyendo que nuestro programa se encontraba sano y salvo en casette o disco.
Sin embargo, la sorpresa es mayúscula cuando comprobamos que dicho programa no se encuentra en el directorio.
Gracias al monitor residente en la Commodore 128 podemos recuperar el programa.
Desde ya no existe ningún remedio cuando, luego de ingresar NEW, se quita la alimentación de la computadora. El harakiri es irremediable!!!

Supongamos tener en memoria el programa:

```
10 REM PGM
20 REM EN LA C-128
30 END
```

y accidentalmente se ingresa NEW.

Tipeen MONITOR (F8). En la pantalla aparecerán los contenidos del contador de programa, acumulador, registros X e Y, stack pointer y del registro de estado.

Ingresen el siguiente comando:

M002D 0035 (return).
Este comando imprime los contenidos de las direcciones hexadecimales comprendidas entre 002D y 0035 (8 bytes).
Sólo observen detenidamente los dos primeros valores, es decir los que corresponden a las direcciones $002D y $002E.
Ellos indican la dirección en donde se comienza a almacenar el programa Basic.
Esta se almacena de la siguiente manera: primero byte bajo y luego byte alto. Generalmente, los contenidos de $002D y $002E son 01 y 1C respectivamente.
Ello indica que la C-128 comienza a almacenar los programas Basic a partir de la dirección $1C01.
Ahora ingresen el comando M1C01 1C20 (return). En la pantalla se imprimirá:

```
1C01 00 00 0A 00 8F 20 50 47
1C09 4D 00 1C 1C 14 00 8F 45
1C11 4E 20 4C 41 40 43 2D 31
1C19 32 38 00 00 00 00 00 00
```

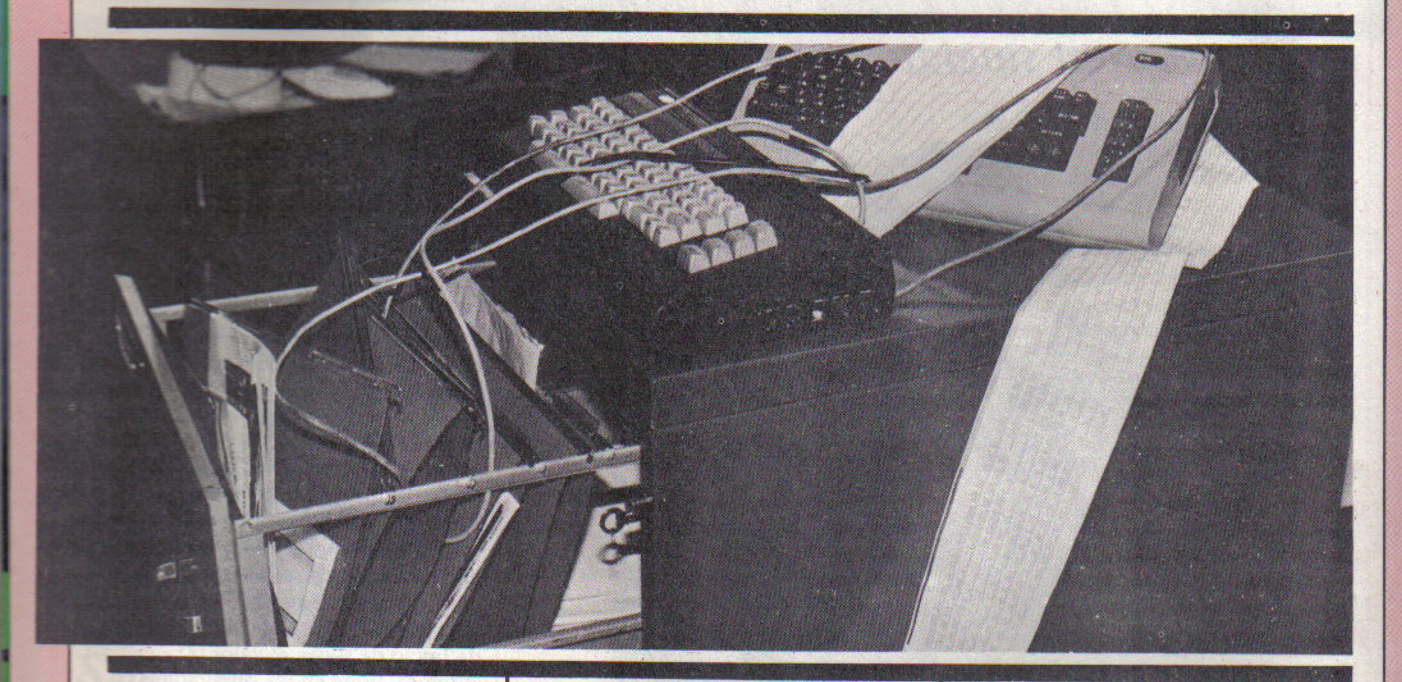
Al lado de cada línea (formada por 8 bytes, 8 valores) se imprimen en video inverso los códigos ASCII de cada uno de los valores almacenados.
Observen como se notan los comentarios de las sentencias REM.
El significado de cada uno de los bytes aquí impresos, se describe en la nota "ALMACENAMIENTOS DE LOS PROGRAMAS" del número anterior.
Como allí mencionamos, el fin de encadenamiento se indica con 00. Al efectuar el comando NEW, el intérprete coloca en las direcciones de inicio de Basic el valor 00.
Además, pone en las direcciones $1210 y $1211 (puntero fin de programa Basic) el valor de $03 y $1C respectivamente.
Para recuperar el programa debemos hacer dos cosas:
1) Cambiar el contenido de las direcciones $1C01 y $1C02 para que apunten hacia la dirección donde comienza la segunda línea, es decir $1C0B.
Esto se logra a través de > $1C01 0B 1C (return).
2) Poner como fin de programa la dirección $1C1C, lo cual se hace a través de > 1210 1C 1C (return).
Si vuelven a tipear M1C01 1C20 (return), verán lo siguiente:

```
1C01 0B 1C 0A 00 8F 20 50 47
1C09 4D 00 1C 1C 14 00 8F 45
1C11 4E 20 4C 41 40 43 2D 31
1C19 32 38 00 00 00 00 00 00
```

Finalmente tipeen X para salir del modo monitor.
Luego hagan LIST y verán el programa nuevamente en memoria.

MANEJO DE ARCHIVOS (1ra. parte)

Para aquellos que tengan la unidad de disco 1541, les comentamos cómo es el manejo de archivos secuenciales y relativos.



Poder agregar a nuestro equipo una unidad de disco, significa aumentar las posibilidades de aplicación.
Con la llegada de la 1541, el trabajo con archivos es mucho más rápido y fiable. Recuerden el tiempo que se tardaba con el datassette.
Tal vez una posible definición de archivo sea "el almacenamiento ordenado de datos". Nosotros podemos diseñar programas que administren correctamente esa información.
Para ello necesitamos un periférico rápido y fiable, como la 1541, aunque algunos digan que es un poco lenta.
Básicamente este periférico puede trabajar con dos tipos de archivos: secuenciales y relativos.
Existe un tercer tipo, denominado Random, que, debido al análisis efectuado, es lo mismo que trabajar con archivos secuenciales.
Cada tipo se caracteriza por el acceso a la información almacenada.
Como decía "Jack el Destripador", vayamos por partes...

Archivos secuenciales

Para tomar un dato, en este tipo de archivos, debemos pasar por los anteriores, confirmando de esta manera el nombre de secuencial.
En general, el procedimiento para trabajar con cualquier tipo de archivos es el siguiente:
1) Abrir el archivo
2) Tomar o poner información.
3) Cerrar el archivo.
Imaginen que, en vez de una computadora y de un drive, tenemos un viejo archivo metálico. Si queremos tomar un dato lo primero que debemos hacer es abrir el "cajón" correspondiente. De otra manera por más que metamos la mano no podremos sacar nada. Una vez abierto, procedemos a buscar lo que queremos. Luego que obtuvimos lo deseado cerramos el cajón para que nadie se lo lleve por delante. Esto ejemplifica los tres puntos arriba citados. Desde ya abrir un archivo desde la computadora no significa que debamos buscar cajones en la consola.
Esta apertura se representa a través de un comando: OPEN.

El cierre a través de CLOSE

Para cada tipo de archivo el formado del OPEN es distinto. Por ejemplo, para abrir un archivo secuencial el formato es:
OPEN N,D,C,"nombre del archivo",S,R"
donde:
N: Es el número de archivo, un valor comprendido entre 0 y 255.
D: Es el dispositivo que vamos a utilizar. Para nuestro caso éste es el número 8 (disketera).
C: Es el número de canal, comprendido entre 2 y 14. El número 1 y el 15 se

utilizan para dialogar con el sistema operativo disco (DOS).
S: Indica el tipo de archivo (S=Sequential).
R: Indica si vamos a leer o a escribir. En este caso la R indica lectura (Read). También puede ser escritura (W=Write).
Un ejemplo en concreto puede ser:
OPEN2,8,2,"MAESTRO,S,R"
Esto indica que abriremos un archivo llamado MAESTRO y que leeremos información.
Algo que no mencionamos es cómo se ponen o se quitan datos desde o hacia un archivo.
Para ello se utilizan las sentencias INPUT# (toma) y PRINT# (pone). Ojo que este PRINT no se puede abreviar con "?".
Si abrimos un archivo para leer debemos usar el INPUT#, mientras que para insertar o para poner datos usamos el PRINT#.
Un ejemplo simple se ilustra en el listado 1. Allí se abre un archivo secuencial (denominado TEST) y ponemos una leyenda.
Noten que el PRINT y el CLOSE están seguidos por el número que corresponde al del archivo antes abierto.
Si hubiésemos puesto otro número en lugar del 4, el intérprete iría a buscar un archivo que nunca fue abierto.
Si, en otro momento, queremos leer la información almacenada en el archivo TEST, debemos hacer lo que se indica en el listado 2.
Si todo sale bien, la variable A$ deberá tener el mensaje antes grabado.
Por supuesto que las aplicaciones que podemos dar pueden ser mucho más complejas que las descriptas hasta aquí. Una de ellas podría ser el almacenar un vector que, por algún motivo, más tarde utilizaremos.
El listado 3 graba en el archivo VECTOR el vector A de 100 elementos.
Cuando debamos recuperar esos valores los leemos como se indica en el listado 4.

Listado 1

```
10 OPEN2,8,2,"TEST,S,W"
20 PRINT#2,"ESTO ES UNA
PRUEBA"
30 CLOSE2
40 STOP
```

Listado 2

```
10 OPEN2,8,5,"TEST,S,R"
20 INPUT#2,A$
30 PRINTA$
40 CLOSE2
50 STOP
```

Listado 3

```
100 REM GRABAMOS VECTOR
110 OPEN2,8,2,"VECTOR,S,W"
120 FORI=1TO100
130 PRINT#2,A(I)
140 NEXTI
150 CLOSE2
160 REM CONTINUA EL
PROGRAMA
```

Listado 4

```
200 REM GRABAMOS VECTOR
210 OPEN2,8,2,"VECTOR,S,R"
220 FORI=1TO100
230 INPUT#2,A(I)
240 NEXTI
250 CLOSE2
260 REM CONTINUA EL
PROGRAMA
```

GRAFICADOR DE FUNCIONES

Tipo: Utilitario
Comp.: Drean Commodore 64
Conf.: Básica
Autor: Roberto Ferrante

En más de una ocasión debemos realizar gráficos de funciones, ya sea para estudiarlos o para efectuar alguna otra tarea.
El programa que aquí les ofrecemos permite graficar funciones utilizando la pantalla en alta resolución. Su autor ha creado técnicas que posibilitan ingresar la función por teclado. De esta manera nos desentendemos del ingreso de la misma directamente en el programa.

Primeramente se nos pide la función a graficar. Luego debemos ingresar los límites de graficación. Para ello el programa posiciona el cursor en cada uno de los ejes. Primero se ingresan los límites sobre el eje Y (positivo y luego negativo). Segundo se ingresan los límites sobre el eje X (primero negativo, luego positivo).
El programa comienza a analizar la función. Es decir, comienza a comprobar que la función es sintácticamente correcta (no le faltan paréntesis, etc.).
En caso de que haya sido escrita en forma incorrecta, el programa imprimirá el mensaje correspondiente. A través de F1 volvemos al menú. Caso contrario comenzará a imprimir la función deseada.
Al finalizar esta tarea, se debe oprimir F1 para volver al menú principal.
El programa utiliza una rutina en lenguaje máquina que contribuye a la graficación y, además, otra rutina que detecta los errores en la función y casos en que se debe dividir por cero. Esto lo decimos para advertirles que la computadora "no queda como antes" luego de cargar y ejecutar este programa. Para subsanar ese inconveniente deben resetear la misma.
Creemos que este soft será de utilidad para los estudiantes de todas las edades.

```
5 DEF FNA(X)=X:REM.................................................
110 POKE49408,112:POKE49409,23
120 GOSUB7010
130 FORI=0TO24
140 READA:POKE40703+I,A:NEXT
150 POKE2053,150
160 GOTO2000
200 POKE150,63:POKE151,63:POKE40704,0:POKE40709,32
210 SYS40703
400 SX=320/(XS-XI)
410 SY=184/(YS-YI)
420 XO=SX*XI*(-1)
430 YO=SY*YS
440 IFXI>0THEN635
450 IFXS<0THEN635
598 IFXI>0THEN635
599 IFXS<0THEN635
600 BI=7-(XOAND7):BL=2↑BI
601 BY=8512+8*INT(XO/8)
602 FORI=0TO22
603 FORJ=0TO7
605 POKEBY+J+320*I,BL
607 NEXTJ
608 NEXTI
635 IFYI>0THEN700
636 IFYS<0THEN700
640 ME=8512+320*INT(YO/8)+YO-8*INT(YO/8)
642 FORI=0TO39:POKEME+8*I,255:NEXT
700 PRINT"J":POKE53272,PEEK(53272)OR8
710 POKE53265,PEEK(53265)OR32
750 POKE40704,240:POKE40709,4:POKE150,232:POKE151,7
760 SYS40703
800 ME=0
810 X=(ME-XO)/SX
```

```
820 AS=YO-FNA(X)*SY
825 IFAS>184THEN840
826 IFAS<0THEN840
830 GOSUB1000
840 ME=ME+1:IFME<320THEN810
845 GOSUB5000
850 GETS$:IFS$=""THEN850
860 IFASC(S$)<>133THEN850
900 PRINT"♥":POKE53272,PEEK(53272)AND247
910 POKE53265,PEEK(53265)AND22
920 GOTO2000
1000 BI=7-(MEAND7)
1001 LI=ASAND7
1002 RO=INT(AS/8)
1003 CH=INT(ME/8)
1010 BY=8512+RO*320+CH*8+LI
1015 IFBY<8512THENGOTO1030
1016 IFBY>15872THENGOTO1030
1020 POKEBY,PEEK(BY)OR2↑BI
1030 RETURN
2000 POKE53280,0:POKE53281,0:PRINT"♥":PRINT"■"
2010 PRINT"┌──────────────────────────────────────┐"
2015 PRINT"|                                      |"
2020 PRINT"|         GRAFICADOR DE FUNCIONES      |"
2030 PRINT"|                                      |"
2040 FORI=1TO2:PRINT"|                                                  |":NEXT
2042 PRINT"|   F(X)=                              |"
2043 FORI=1TO2:PRINT"|                                                  |":NEXT
2044 FORI=1TO6:PRINT"|                   |                              |":NEXT
2045 PRINT"|        ──────────┼──────────         |"
2046 FORI=1TO6:PRINT"|                   |                              |":NEXT
2050 PRINT"└──────────────────────────────────────┘"
2052 PRINT"IIIIIIIIIIIIIIIII▶▶▶▶▶▶▶▶▶";
2053 GOSUB 4000
2054 A$=Q$
2100 ME=2061:FORI=1TO LEN(A$)
2110 B$=MID$(A$,I,1)
2120 IFABS(ASC(B$)-75)>10THENGOTO2190
2130 IFB$="T"THENAS=192:I=I+2:GOTO2260
2150 IFB$="E"THENAS=189:I=I+2:GOTO2260
2160 IFB$="L"THENAS=188:I=I+2:GOTO2260
2170 IFB$="S"THENIFMID$(A$,I+1,1)="I"THENAS=191:I=I+2:GOTO2260
2180 IFB$="S"THENIFMID$(A$,I+1,1)="Q"THENAS=186:I=I+2:GOTO2260
2190 IFB$="+"THENAS=170:GOTO2260
2200 IFB$="-"THENAS=171:GOTO2260
```

PROGRAMAS

```
2210 IFB$="%"THENAS=172:GOTO2260
2220 IFB$="/"THENAS=173:GOTO2260
2230 IFB$="↑"THENAS=174:GOTO2260
2240 IFB$="π"THENAS=255:GOTO2600
2250 AS=ASC(B$)
2260 POKEME,AS
2270 ME=ME+1
2280 NEXT I
2290 POKEME,58
2300 POKEME+1,143
2310 X=1:AS=FNA(X)
2500 PRINT:PRINT"▓▓▓▓▓▓▓▓▓▓▓▓▓▓▓▓▓▓▓▓▓";:GOSUB4000
2510 YS=VAL(Q$)
2520 PRINT:PRINT"▓▓▓▓▓▓▓▓▓▓▓▓▓▓▓▓▓▓▓▓▓▓▓▓▓▓▓▓▓▓▓▓▓▓";:GOSUB4000
2530 YI=VAL(Q$)
2540 PRINT:PRINT"▓▓▓▓▓▓▓▓▓[IIIII]";:GOSUB4000
2550 XI=VAL(Q$)
2560 PRINT:PRINT"]▓▓▓▓▓▓▓▓▓▓▓▓▓▓▓▓▓▓▓▓▓▓▓▓▓▓▓▓";:GOSUB4000
2570 XS=VAL(Q$):IFXS=0THENXS=.0000001
2580 GOTO200
4000 Q$=""
4010 PRINT"▓ ";:PRINT"▓";:GETS$:PRINT"▓ ";:PRINT"▓";:IFS$=""THEN4010
4015 IFASC(S$)=13THENRETURN
4016 IFASC(S$)=20THENQ$=LEFT$(Q$,LEN(Q$)-1):PRINT'▓";:GOTO4010
4020 Q$=Q$+S$:PRINTS$;
4030 GOTO4010
5000 POKE56334,PEEK(56334)AND254:POKE1,PEEK(1)AND251
5003 Q$="F(X)="+A$
5005 FORI=1TOLEN(Q$)
5010 ME=ASC(MID$(Q$,I,1))-64:IFME<0THENME=ME+64
5020 FORJ=0TO7
5030 BY=PEEK(53248+8*ME+J)
5040 POKE8536+J+8*(I-1),BY
5050 NEXTJ
5060 NEXTI
5070 POKE1,PEEK(1)OR4:POKE56334,PEEK(56334)OR1
5080 RETURN
6000 REM ERRORES
6010 ER=PEEK(49920):IFER=20THEN840
6020 IFER=17THEN840
6021 IFER<>14THEN6030
6022 POKE53272,PEEK(53272)AND247:POKE53265,PEEK(53265)AND22
6023 PRINT"[IIIIIIIIII]"
6024 PRINT"        ┌─────────────────────┐"
6025 PRINT"        │  DATOS INCOMPATIBLES │"
6026 PRINT"        └─────────────────────┘"
6027 GETQ$:IFQ$=""THEN6027
6028 GOTO2000
6030 IFER<>11THENPRINT:PRINT:PRINT:PRINT"ER=";ER:PRINT"EN=";PEEK(49921)+256*PEEK
(49922):END
6040 PRINT"  INCORRECTA":PRINT"]";
6045 GETQ$:IFQ$=""THEN6045
6047 PRINT"▓▓▓▓▓▓▓▓▓                          ":PRINT"]";
6050 PRINT"▓▓▓▓▓▓▓▓▓";:GOTO2053
7000 REM ON ERROR
7010 F=0:FORI=49152TO49218:READA:POKEI,A:F=F+A:NEXTI
7020 IFF<>8181THENPRINT"ERROR EN DATAS 7030-7070. ":STOP
7030 DATA169,11,141,0,3,169,192,141,1,3,96
7040 DATA224,128,208,9,76,139,227,234,234,234,234,234,234
7050 DATA142,0,195,165,57,141,1,195,165,58,141
7060 DATA2,195,173,0,193,133,20,173,1,193,133
7070 DATA21,32,19,166,230,95,230,95,230,95,165,95
7080 DATA133,122,165,96,133,123,108,8,3
7090 SYS49152:RETURN
8000 DATA 169,0,160,0,153,0,32,200,208,3,238,5,159,174,5,159,228,151,208,240
8010 DATA 196,150,208,236,96
```

EL CHIP TED

Comentamos las características sobresalientes de uno de los "órganos" más importantes de la C-16.

El circuito integrado TED, el cual viene incluido dentro de la C-16 y de la PLUS 4, es el circuito más importante del equipo.

Prácticamente realiza todas las operaciones, como ser generación de sonido, generación de color, y administración de memoria, entre otras cosas.

Más exactamente, el TED realiza:

- -Refresco dinámico de RAM
- -Control programable para CRT (sistema PAL o NTSC)
- -Display de 40 columnas por 25 líneas
- -Generación de 121 colores diferentes
- -Flash programable
- -Video inverso programable
- -Cursor "hardware"
- -Posibilidad de desconectar la pantalla para que la CPU procese más rápido.
- -Cinco modos de pantalla
- -Expandible internamente
- -Tres relojes de generación de interrupción (internos).

El TED es un dispositivo de 48 pines que controla la salida de video, el sistema de reloj, el control dinámico de RAM, la selección de ROM y el control de teclado.

Además contiene 34 registros de control, los cuales pueden ser accedidos por el usuario ya que están en memoria RAM.

Modos de impresión

En los tres modos de impresión, este circuito integrado imprime 25 líneas que contienen 40 caracteres cada una.

Cada caracter impreso puede tener uno de los 16 colores juntamente con uno de los 8 posibles niveles de luminancia.

El puntero del caracter dentro de la matriz de video determina el caracter que será impreso en una determinada posición de esa matriz.

Cada una de estas posiciones está asociada a un byte (llamado byte de atributo) que determina el color del caracter, el nivel de luminancia y si será impreso con o sin flash.

El TED busca los punteros de caracter en un área de memoria llamada matriz de video y la información del color en el área de atributos.

La matriz de video consiste en un área de memoria formada por 1000 direcciones, cada una de las cuales contiene el puntero del caracter.

Dentro del TED, la dirección de la matriz de video está determinada por el registro de matriz de video (bits 3,4,5,6 y 7 del registro número 20 del TED), el cual provee los cinco bit más significativos de la dirección de la matriz de video.

De esta manera se pueden obtener 32 posibles direcciones de inicio de la matriz de video.

La tabla 1 las representa acorde a los valores de los bits 3,4,5,6 y 7 del registro 20, que más tarde explicaremos.

Cada byte en la matriz de video es utilizado para señalar al actual los datos que forman a un determinado caracter.

El área correspondiente a los atributos de cada caracter también consiste en 1000 direcciones consecutivas de memoria.

Contiene el bit de flash, los cuatro bits para el color del caracter, y los tres bits para el nivel de luminancia.

La dirección de inicio de esta área es también controlada por el registro de matriz de video.

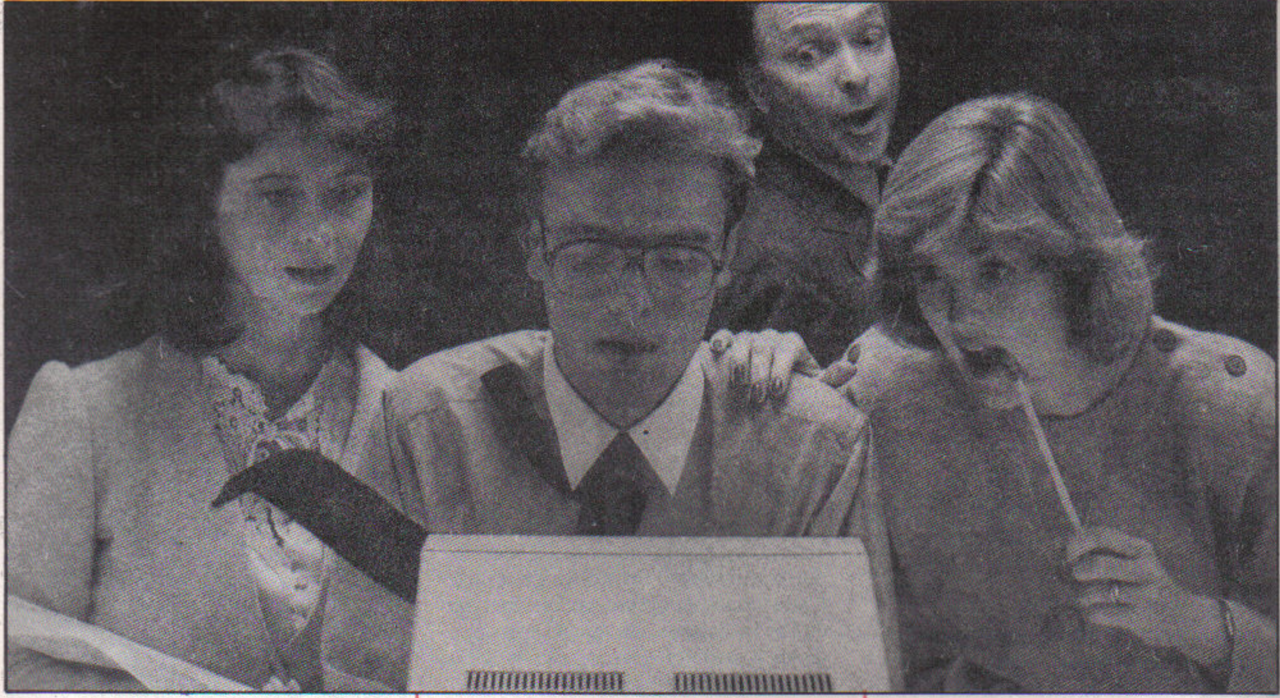
En el modo estándar de impresión, los caracteres son impresos con 8 puntos de ancho por 8 de altos. Esta "pequeña" matriz se llama Dot y hace referencia a la constitución del caracter.

Tabla 1

Bits	Dirección de inicio
00000	$0400
00001	$0C00
00010	$1400
00011	$1C00
00100	$2400
00101	$2C00
00110	$3400
00111	$3C00
01000	$4400
01001	$4C00
01010	$5400
01011	$5C00
01100	$6400
01101	$6C00
01110	$7400
01111	$7C00
10000	$8400
10001	$8C00
10010	$9400
10011	$9C00
10100	$A400
10101	$AC00
10110	$B400
10111	$BC00
11000	$C400
11001	$CC00
11010	$D400
11011	$DC00
11100	$E400
11101	$EC00
11110	$F400
11111	$FC00

SISTEMAS EXPERTOS

Las máquinas "inteligentes" están siendo actualmente utilizadas para efectuar diagnósticos y resolver problemas tanto en la medicina como en la industria. Incluso algunos de los productores de software están haciendo experiencias en este campo.



Si definimos a un "experto" como alguien que sabe más que la mayoría de las personas acerca de un determinado tema, veremos que, prácticamente, todos los días de la semana recurrimos a los consejos de un experto.
Si estamos enfermos recurrimos a un médico, quien luego de analizarnos y de formularnos una serie de preguntas llegará a un diagnóstico y recomendará un tratamiento.
Si nuestro automóvil comienza a fallar, más que seguro lo llevaremos al taller donde el mecánico nos dirá el arreglo que hay que efectuar.
Por otro lado, si alguien formula falso testimonio sobre nuestra persona, recurriremos a un abogado para ver si podemos o no iniciarle juicio.
Desde ya, si nosotros confiamos en estos expertos es porque consideramos que ellos tienen un conocimiento más que suficiente de un determinado tema.
La informática hoy nos ofrece algo mucho más interesante y mucho más barato: los sistemas expertos.
Estos son programas diseñados para actuar como consultores en áreas tales como finanzas personales o en el cuidado de la salud.

Ahora bien, ¿son realmente sustitutos de los profesionales expertos? De acuerdo con quienes los producen la respuesta sería negativa, ya que ellos insisten en que los programas son solamente consultores y que, de todos modos, habrá que recurrir a la ayuda de los verdaderos profesionales.

Pero no está muy lejos el día en que un nuevo tipo de programa reemplazará a los expertos, o al menos llevará a cabo buena parte de lo que ellos realizan.

Estos sofisticados programas contienen datos básicos de conocimiento que sólo podrían ser adquiridos por los expertos humanos a través de varios años de estudio.

Y lo que es aún más, los sistemas expertos más avanzados que se están desarrollando incorporan incluso algunas reglas de lógica y análisis que los profesionales combinan con su archivo de datos para resolver los problemas que se presentan.
Un ejemplo en concreto de la aplicación de los sistemas expertos es el análisis de datos geológicos para detectar manchas de petróleo, labor que anteriormente era realizada por geólogos e ingenieros.
Algunas personas creen que estos sistemas serán algo corriente en la próxima generación de computadoras hogareñas llevando a nuestra casa, con sólo oprimir un botón, el consejo del médico o de otros profesionales.
Otros, en cambio, advierten que la aplicación prematura de estas innovaciones puede traer serios trastornos, especialmente si esos sistemas están basados en una concepción errónea acerca del proceso de toma de decisiones.
A pesar de que el desarrollo de

sistemas expertos está todavía en pañales, la cuestión abre otro capítulo en el debate referente a la inteligencia artificial.
Hace algunos años atrás, Joseph Weizenbaum, profesor de computación en el Instituto de Tecnología de Massachusetts (MIT) realizó un programa al cual tituló "Eliza". Intentaba demostrar cómo una computadora actúa como un psicólogo. Eliza hacía una serie de preguntas al usuario sobre cómo se sentía, luego seleccionaba algunas palabras claves o frases de las respuestas para guiar su "terapia".
Hay quienes consideran a Eliza como un programa pionero de los sistemas expertos.
"Yo ni había oído mencionar ese término cuando estaba desarrollando el programa", dice ahora Weizenbaum.
Algunos científicos sostienen que el desafío de diseñar un sistema experto consiste en definirlo; es decir que se supone qué es y cómo debe funcionar. Pero no todo el mundo científico está de acuerdo con este "desafío". Por ejemplo, Weizenbaum piensa que Eliza es un sistema experto porque él consultó a expertos antes de realizar el programa.
Aunque Eliza parece estar realmente escuchando y respondiendo, el programa sólo sigue una serie de reglas dadas por Weizenbaum.
Si uno dice que tuvo un mal día, el programa nos pedirá que relatemos lo sucedido. Luego puede preguntar cómo nos afectan determinados hechos, o qué creemos nosotros acerca de tal o cual cosa.
Eliza es más una especie de diario interactivo que un experto.
El término "experto" se aplica con mayor precisión a sistemas que funcionan en forma experta.
Pero eso es todavía muy ambiguo, dice Weizenbaum. Añade que "si uno caracteriza como sistemas expertos aquel que funciona expertamente, muchos de los programas científicos y técnicos que se han realizado y que realizan una labor casi perfecta en las áreas en donde se aplican, deberían ser considerados todos como sistemas expertos. Por lo tanto el término no es lo suficientemente preciso".
"Aquí hay un ejemplo de algo que nadie considera un sistema experto: Hoy en día casi todos los aterrizajes aéreos son realizados por una computadora de abordo".
"A menudo me pregunto —continúa diciendo Weizenbaum— qué hubiese pasado si el programa que controla todos los movimientos del avión para realizar el aterrizaje hubiese sido desarrollado en los laboratorios del MIT o de la Universidad de Standford. No creo que nos hubiésemos enterado de su fin. Pero en verdad fue hecho, podríamos decir, anónimamente. No tengo idea de quién lo hizo, y sin embargo, hace una tarea que le lleva años de entrenamiento a un ser humano. Sin embargo ésto no es considerado un sistema experto, es extraño".
Por lo tanto, definirlo no es tan simple como hacer referencia a una computadora que reemplaza las actividades de un humano.
Las máquinas han estado haciendo éso por años. Por ejemplo, a pesar de no ser catalogadas por académicos como sistemas expertos, las computadoras encargadas de controles de proceso llevan a cabo funciones previamente desempeñadas por hombres especialmente entrenados.
"Actualmente —dice Weizenbaum—, uno puede recorrer las instalaciones de una destilería de petróleo y encontrarse con muy pocos humanos.
Prácticamente todas las tareas son efectuadas por computadoras".
"Es decir que existe todo un mundo de control computarizado que se ha venido desarrollando durante años y que no se consideran sistemas expertos".
En cambio, los verdaderos sistemas expertos parecen estar definidos de acuerdo con su evolución y arquitectura como un conjunto de datos básicos de reglas y mecanismos de inferencia.
Las computadoras que controlan procesos fueron desarrolladas de otro modo. "Hay muchísimas aplicaciones de procesos de control que han sido muy bien hechas, y que hoy día, a la luz de la nueva concepción de sistemas expertos habrían sido abordados en forma diferente", continúa diciendo Weizenbaum.
El límite entre lo que son los sistemas expertos y la inteligencia artificial es aún más confuso. Para algunos hay una clara diferencia. Para otros un sistema experto que funciona perfectamente implica inteligencia artificial.
Parte del problema está en que los investigadores de inteligencia artificial difieren acerca de cómo enfocar estos temas.
"Desde hace mucho tiempo —dice Weizenbaum—, se perfilan dos concepciones diferentes".
La primera, considera la inteligencia artificial como una rama de la psicología. Esto es usar la computadora para comprender las operaciones de la mente humana a través de programar tareas de alto nivel del mismo modo en que pensamos que la mente humana podría hacerlas.
La otra concepción se basa en la programación de tareas muy complicadas que requieren de la inteligencia humana, pero llevadas a cabo de tal forma que no podrían ser consideradas para ser hechas por el hombre (tampoco sería posible).
Estas dos escuelas de pensamiento se conocen como teóricas y de performance o actuación. Weizenbaum da un ejemplo de la teórica:
"Desde muy temprano la gente se interesó en la idea de las jugadas de ajedrez. Se pensó que si podíamos saber lo que sucedía en la mente del jugador de ajedrez y podíamos de algún modo programarlo en la computadora, no sólo obtendríamos una máquina que jugase bien al ajedrez sino que, además, aprenderíamos mucho acerca de la mente humana, su psicología y su funcionamiento".
Ese fue el comienzo, pero enseguida surgió la tentación de tomar atajos, de sacar partido de alguna característica de las computadoras que nadie creía que tuviera la mente humana.
"Por lo tanto desde el comienzo la tentación no pudo ser resistida, y la gente programó juegos de ajedrez que aprovechaban todas las ventajas de las peculiaridades de las computadoras pero dejaban de lado toda consideración acerca de cómo lo haría la mente humana".
"Así es como hoy en día tenemos poderosas computadoras que juegan increíblemente bien al ajedrez pero que no enseñan nada del pensamiento humano. Tenemos aquí como se comenzó desde la concepción teórica y se terminó tratando el tema desde la performance o actuación".
"Y debido a varias circunstancias, incluyendo el interés y la financiación militar de investigación de la inteligencia artificial desde el punto de vista de la performance, hoy es muy poco lo que se hace desde el enfoque teórico", dice Weizenbaum.
En la Universidad de San Diego, en California, en un centro de investigación llamado Instituto de Ciencia Cognitiva, Paul Smolensky, uno de los investigadores, se ha dedicado fundamentalmente al estudio de modelos matemáticos de aprendizaje, procesos de memoria y de resolución de situaciones concretas.
Utilizando lo que comunmente se acepta como características generales del cerebro humano, el trabajo de Smolensky se centra en un área primaria: el entendimiento de los seres humanos y cómo educarlos, y el conocimiento avanzado en el campo científico.
Uno de los resultados de la

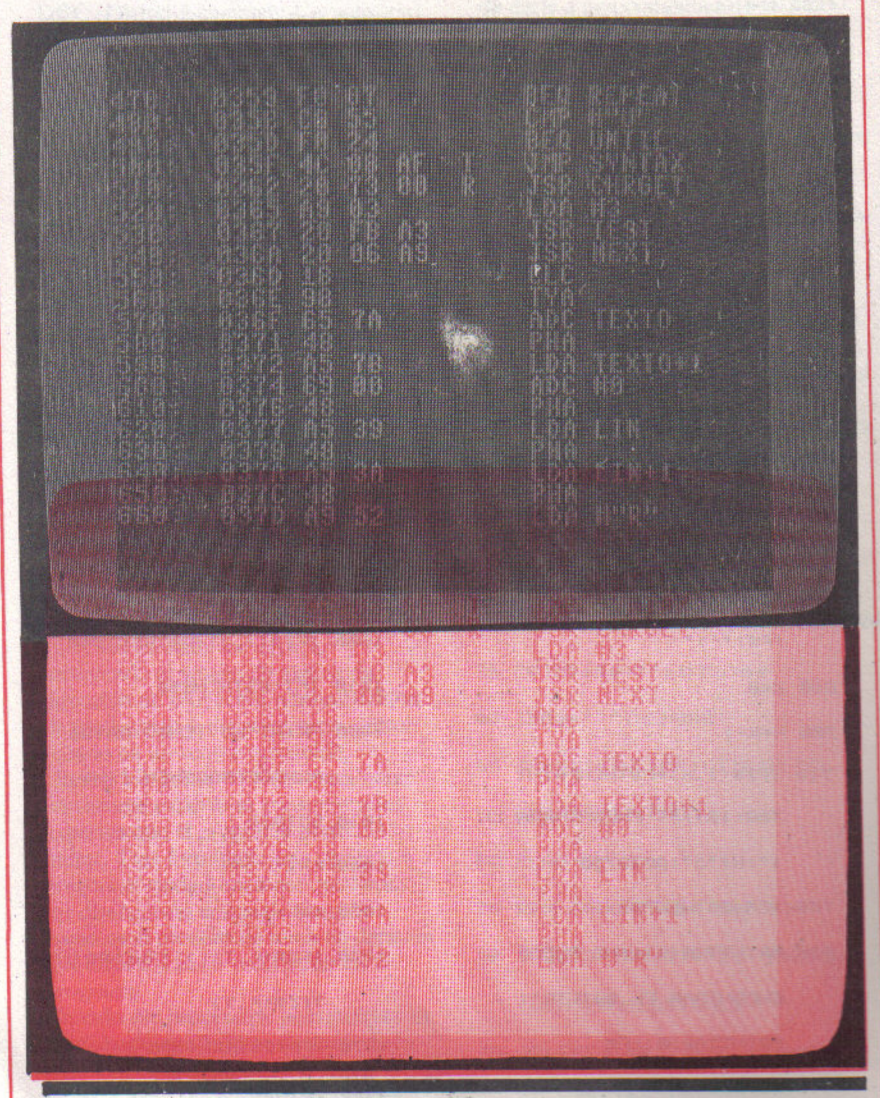
investigación es la sugerencia de la creación de un nuevo tipo de computadoras que, por ejemplo, conecte a varios procesadores entre sí, trabajando paralelamente, como si fuesen neuronas cerebrales.
En la actualidad hay muy pocas máquinas con estas características.
"Hay una idea platónica de lo que debe ser un sistema experto y al mismo tiempo hay un montón de sistemas funcionando que han sido rotulados como tales. Yo —continúa diciendo Smolensky— no creo que realmente se trate de sistemas expertos excepto uno bien conocido por los que estamos en Ciencias de la Computación, el de DEC (Digital Equipment Corporation) realizado para diseñar las instalaciones en sus sistemas de computación VAX".
Este sistema experto, llamado R1/XCOM, fue diseñado por el Dr. John McDermott, director asociado del departamento de Computación y principal investigador del área en la Universidad de Carnegie-Mellon.
Está formado por un sistema de minicomputadoras VAX. A través de él, DEC se ahorra 2.5 millones de dólares anuales en costo de comercialización.
R1/XCOM toma apenas un minuto para hacer el procesamiento que a un humano le demanda una hora.
McDermott, junto a ingenieros, científicos y programadores, han formado una corporación llamada Carnegie Group, cuya finalidad es diseñar y comercializar sistemas basados en inteligencia artificial. Este grupo está buscando mercado en las diferentes áreas que puedan interesarse en el uso de sistemas expertos, incluyendo el diseño de ingeniería, dirección de proyectos, administración de producción y diagnóstico y control basado en un sensor mecánico.
Uno de los primeros pasos para crear un sistema experto es consultar a los expertos que el programa pretende imitar. Tras formularle una serie de preguntas cuidadosamente detalladas, los diseñadores tratan de descubrir cómo es el proceso de toma de decisión que ellos tratan de reconstruir en el programa.
Cuando ese proceso de pensamiento es combinado con la base de datos de la realidad, el sistema experto ideal debería tener la capacidad necesaria como para analizar la información recibida y llegar a la solución correcta.
Este es un punto vital para los críticos de sistemas expertos e inteligencia artificial. Por ejemplo, si nosotros le pedimos a alguien que nos comente de qué se trata la película "Juegos de Guerra" éste nos dirá, más o menos, lo siguiente: "Es acerca de un chico que interfiere el sistema nacional de defensa con su computadora hogareña y que casi desata una guerra nuclear".
Pero el sistema de defensa no estaba expuesto a esta vulnerabilidad hasta que el gobierno Norteamericano decidió que los seres humanos no eran lo suficientemente confiables como para manejar los códigos necesarios que ocasionarían el disparo de armas nucleares.
Por lo tanto esas armas fueron puestas bajo el control de computadoras, las cuales no fallarían, por problemas emocionales, en el momento crucial.
"Debe ser el juicio humano el que debe decidir acerca de la conveniencia o no de darle a una computadora un cierto rol en un sistema de toma de decisiones", dice Smolensky.
Las computadoras son capaces de llevar a cabo tareas antes desempeñadas por seres humanos, pero eso no las convierte en seres inteligentes.
El conocimiento de los expertos deriva en forma significativa de la intuición y de los procesos intuitivos.
Los expertos no son absolutamente concientes de ello y probablemente no hagan referencia a su intuición cuando son interrogados acerca de cómo hacen lo que hacen.
Y si queremos entender la capacidad de un experto, tenemos que entender la intuición.
Smolensky formula una advertencia sobre el hecho de utilizar demasiada tecnología en muy poco tiempo, especialmente en áreas que tienen un efecto directo en la vida humana.
El señala que inclusive ocurren errores no previstos cuando se instala un sistema (computadoras más programas) en una oficina.
Agrega que gracias a los seres humanos es posible arreglar esos errores y hacer que todo marche bien, evitando que ocurran desastres históricos.
"Si enfocamos el problema de tomar decisiones en forma inteligente, como algo que sólo podemos comprender cuando comprendamos el rol de la intuición, y si uno acepta que la intuición no la vamos a comprender durante mucho tiempo, aceptaremos que no podemos seguir dando a las computadoras el poder de tomar decisiones importantes".
A continuación les dejamos el listado de un programa que realiza deducciones acorde a los datos ingresados.
El método básicamente consiste en silogismos y, ante su sorpresa, les dará conclusiones increíbles.
Carguen y ejecuten el programa.
Primeramente aparecerá un signo de pregunta. Aquí ingresen:
CARLOS ES BUENO
Opriman Return e ingresen:
UN BUENO ES RESPONSABLE
y luego tipeen
ES CARLOS RESPONSABLE
Así verán la respuesta.

```
10 REM SILOGISMOS
20 GOSUB1100:REM INICIALIZACION
30 PRINT:A$="":INPUTA$
40 IFA$="?"THEN960
50 IFA$=""THENPRINT"FIN DEL
   PROGRAMA":END
60 FLAG=0
70 REM COMENZAMOS PROCESO
80 IFLEFT$(A$,3)="ES "THEN510
   :REM CONCLUSIONES
30 IFLEFT$(A$,3)="EL "THEN
   A$=MID$(A$,4)
100 IFLEFT$(A$,3)="LA "THEN
    A$=MID$(A$,4)
110 IFLEFT$(A$,3)="UN "THEN
    A$=MID$(A$,4)
120 IFLEFT$(A$,4)="UNA "THEN
    A$=MID$(A$,5)
130 X=LEN(A$)
140 N=0
150 N=N+1
160 IFMID$(A$,N,1)=" "THEN
    B$=LEFT$(A$,N-1):GOTO190
170 IFN<XTHEN150
180 PRINT"NO COMPRENDO":
    GOTO30
190 K=4
200 IFMID$(A$,N+1,1)="F"THEN
    K=5
210 C$=MID$(A$,N+K):REM
220 REM DETECTA 1 CARACTER
230 IFLEFT$(C$,4)="UNA "THEN
    C$=MID$(C$,5)
```

```
240 IFLEFT$(C$,3)="UN "THEN
    C$=MID$(C$,4)
250 IFLEFT$(C$,3)="EL "THEN
    C$=MID$(C$,4)
260 IFLEFT$(C$,3)="LA "THEN
    C$=MID$(C$,4)
270 REM INICIAMOS
280 REM BUSQUEDA
290 N=0
300 N=N+1
310 IFZ$(1,N)=B$THEN350:REM
320 IFZ$(1,N)=""THENZ$(1,N)=B$
    :GOTO350
330 IFN<25THEN300
340 PRINT"NO ME QUEDA ESPACIO
    PARA ALMACENAR MAS SUJETOS"
350 REM
360 REM
370 K=0
380 K=K+1
390 IFZ$(K,N)=C$THEN430:REM
400 IFZ$(K,N)=""THENZ$(K,N)=C$
    :GOTO430
410 IFK<25THEN380
420 PRINT"NO TENGO MAS ESPACIO
    PARA ALMACENAR OBJETOS
    DIRECTOS
430 IFFLAG=1THENPRINTTAB(6);
    "> DE ACUERDO":GOTO30:REM
440 REM
450 FLAG=1
460 M$=B$
470 B$=C$
480 C$=M$
490 GOTO280
500 REM
510 REM
520 REM
530 A$=MID$(A$,4):REM
540 IFLEFT$(A$,4)="UNA "THEN
    A$=MID$(A$,5):REM
550 IFLEFT$(A$,3)="UN "OR
    LEFT$(A$,3)="EL "OR
    LEFT$(A$,3)="LA "
    THENA$=MID$(A$,4)
560 REM
570 X=LEN(A$)
580 N=0
590 N=N+1
600 IFMID$(A$,N,1)=" "THEN
    F$=LEFT$(A$,N-1):GOTO630
610 IFN<XTHEN590
620 PRINTTAB(6);"> NO COMPRENDO"
    :GOTO30
630 REM
640 S$=MID$(A$,N)
650 IFLEFT$(S$,4)=" UN "THEN
    S$=MID$(A$,N+4):GOTO680
660 IFLEFT$(S$,5)=" UNA "THEN
    S$=MID$(A$,N+5):GOTO680
670 S$=MID$(A$,N+1)
680 PRINTTAB(9);"ESTOY BUSCANDO '"
    ;F$;"')"
690 X=0
700 X=X+1
710 IFZ$(1,X)=F$THENPRINT
    TAB(10);"( ENCONTRADA EN
    1";X;")":GOTO750
720 IFX<25THEN700
730 PRINTTAB(6);"NO PUEDO
    ENCONTRAR EL SUJETO":
    PRINTTAB(8);F$
740 GOTO30
750 Y=1
760 Y=Y+1
770 IFZ$(Y,X)=S$THENPRINT
    TAB(6);"> SI":GOTO30
780 IFY<25THEN760
790 Y=1
800 Y=Y+1
810 P$=Z$(Y,X)
820 M=0
830 M=M+1
840 IFZ$(1,M)=P$THEN880
850 IFM<25THEN830
860 IFY<25THEN800
870 PRINTTAB(6);"> NO":GOTO3
880 Q=1
890 Q=Q+1
900 IFZ$(Q,M)=S$THENPRINT
    TAB(6);"> SI":GOTO30
910 IFQ<25THEN890
920 IFM<25THEN830
930 GOTO870
940 REM COMPROBAMOS
950 REM DATOS A
960 REM COMPROBAR
970 INPUT"QUE SUJETO SE VA A
    COMPROBAR";H$
980 T=0
990 T=T+1
1000 IFZ$(1,T)=H$THEN1040
1010 IFT<25THEN990
1020 PRINT"NO TENGO DATOS
     ALMACENADOS EN ";H$
1030 GOTO30
1040 K=1
1050 K=K+1
1060 IFZ$(K,T)<>""THENPRINTK;
     T;Z$(K,T)
1070 IFK<25THEN1050
1080 GOTO30
1090 REM INICIALIZAMOS
1100 REM VARIABLES Y MATRIZ
1110 PRINTCHR$(147)
1120 DIMZ$(25,25)
1140 RETURN
```

LOS MNEMOTECNICOS EN ACCION

Iniciamos la última fase de esta serie de notas comentándoles la ventaja de trabajar con los mnemotécnicos.



En números anteriores hemos dicho que antes de explicar como se programa en Assembler debíamos decir cómo se hace en código de máquina. Respetando ese criterio comentamos cómo son los direccionamientos, incrementos y decrecimientos, saltos relativos, saltos incondicionales y saltos a subrutinas.

A partir de este número explicaremos cómo se realizan estas tareas pero utilizando los mnemotécnicos.

El **direccionamiento inmediato**, es decir donde el operando está a continuación del código de operación, se realiza a través de:

LDA #$AA

El símbolo "#" denota que se trata de ese direccionamiento. En este caso se carga el acumulador con $AA.

También pudimos haber cargado el registro X o Y con el mismo procedimiento. Es decir:

LDX #$AA; carga el reg. X con $AA

LDY #$FF; carga el reg. Y con $FF

El formato general es:

LDA #$XX; XX es un valor hexadecimal válido.

El **direccionamiento página cero** se hace a través de:

LDA $C0

Aquí se carga el acumulador con el contenido de la dirección $00C0.

Como estamos en los primeros 256 bytes de memoria, no hace falta especificar la parta alta ya que ésta es siempre cero.

Noten que no se pone el símbolo "#". De esta manera, cuando se traduzca la instrucción, el compilador entenderá que es una dirección y no un operando.

Claro que también pudimos haber puesto:

LDA $00C0

La única diferencia que hay entre éste y aquél es que el último demandará más tiempo en ejecutarse (recuerden que todo lo que esté en la página cero es más veloz).

Se puede hacer el mismo procedimiento para los registros X e Y.

Es decir:

LDX $C0; carga reg. X con el contenido de la dirección $C0

LDY $0F; carga reg. Y con el contenido de la dirección $0F

El formato general de la instrucción es, tomando como ejemplo la carga del acumulador:

LDA $XX; donde XX es una dirección válida.

El **direccionamiento indexado página** cero, es de la forma:

LDA $C0,X

Aquí se cargará el acumulador con el contenido de la dirección dada por $C0 más el contenido del registro X, es decir ($C0+(X))

Como recordarán, en el **direccionamiento absoluto** la dirección efectiva (donde se encuentra el operando) se colocaba a continuación del código de operación.

A través de los mnemotécnicos, ésto se hace como:
LDA $32E4
La cual cargará el acumulador con el contenido de la dirección $32E4.
También es posible cargar el registro X e Y utilizando este modo.
Para el **direccionamiento indexado**, el proceso es:
LDA $C000,X
En este caso el acumulador se cargará con el contenido de la dirección $C000 más el contenido del registro X, ($C000+X)→Acc.
De la misma manera se pudo haber usado como registro índice el Y.
Continuamos con el **direccionamiento indirecto indexado (post indexado)**.
Este se representa como:
LDA ($C0),Y
El acumulador se carga con el contenido de la dirección cuyo byte bajo está dado por el contenido de la dirección $C0 y, a éste, sumándole el contenido del registro Y, es decir ($C000)+(Y), y cuyo byte alto es el contenido de la dirección $C1.
Si hay carry en esa suma, éste se transfiere a la dirección siguiente; a la $C1.

El registro índice que interviene con este direccionamiento es siempre el Y.
El **direccionamiento indexado indirecto (pre indexado)** se representa como:
LDA ($C0,X)
El acumulador se cargará con el contenido de la dirección cuyo byte bajo está dado por el contenido de la dirección $C0 más el contenido del registro X, es decir ($C0+(X)).
El byte bajo de la dirección está dado por ($C1).
Si hay carry en la suma, éste no se transfiere a la dirección siguiente.
Los dos últimos direccionamientos utilizan, siempre, direcciones en página cero, para luego acceder a la dirección efectiva.
Sería muy útil que miren nuevamente el set de instrucciones del micro. Allí, cada instrucción, tiene con qué modos de direccionamiento puede utilizarse.
Para poder trabajar en Assembler necesitamos algún editor como lo son, por ejemplo, el HESMON o el 64MON.
Cada uno de ellos dispone de comandos muy similares. Por ejemplo para entrar una instrucción en mnemotécnico se usa el comando "A" seguido por la dirección en donde queremos ubicarla y, finalmente, el mnemotécnico.
Por ejemplo:
A C000 LDA #$FF
A C002
Cada vez que ingresamos una instrucción, ésta es interpretada a su correspondiente en código de máquina.
Luego se incrementa la dirección inicial acorde a la instrucción tipeada.
A continuación realizaremos un pequeño programa en Assembler, el cual carga el registro X con $FF y luego transfiere su valor al acumulador:
A C000 LDX #$FF
A C002 TXA
A C003 BRK
Para ejecutarlo debemos utilizar el comando G (GO) de la siguiente manera:
G C000
Los usuarios de la C-16 y C-128 no tendrán mayores problemas debido a que sus máquinas disponen de un monitor residente.
Todo lo hablado hasta aquí se puede aplicar en cualquier equipo Commodore.

ALMACENAMIENTO HORARIO

```
10 REM ********************
20 REM PRUEBA DE
30 REM ALMACENAMIENTO HORARIO

LIST

10 REM 05/09/86 05:01 PM
20 REM PRUEBA DE
30 REM ALMACENAMIENTO HORARIO
READY.
```

Tipo: Utilitario
Comp.: Drean Commodore 64
Conf.: C-64 y/o Drive compatible

A través de este programa utilitario podrán conocer cuál fue la última versión de sus programas.
Cada vez que ellos sean listados en pantalla, se imprimirá la fecha y la hora actual, datos que oportunamente fueron ingresados por nosotros.
En caso de haber tipeado los valores correspondientes a la sentencia DATA correctamente, el programa nos preguntará por la fecha actual. La debemos ingresar respetando el formato pedido: primero el mes, luego el día y finalmente el año, y todos ellos separados por un signo "/". En caso de no respetar este formato, el programa interrogará nuevamente.
Luego se nos pedirá el ingreso del tiempo actual. El formato pedido es primero la hora y, seguido, los minutos. Ejemplo de ello es 0921, que significa 9 horas y 21 minutos.

Debido a las características del reloj interno de la C-64, la hora ingresada es del tipo AM y PM. Es decir que una hora como las 22.45 debe ser ingresada como 10.45.

Seguidamente el programa pedirá saber si es AM (A) o PM (P).

Así conseguimos poner en hora el reloj interno.

Para aquellos que supongan que el programa trabaja con TI$, les decimos que están equivocados.

Como mencionamos anteriormente, la C-64 dispone de un reloj de tiempo real el cual puede ser puesto en hora y, además, permite seleccionar la hora de alarma.
Les aconsejamos que antes de ejecutar el programa hagan una copia en disco ya que este se autoborra de la memoria.
Antes de iniciar la puesta en hora, el programa imprime los datos tipeados por nosotros pidiéndonos la confirmación de los mismos. Así podemos evitar el ingreso de un dato erróneo.
Si nuestros datos fueron los correctos, el programa pondrá en hora el reloj y se autoborrará de la memoria.
A partir de aquí ya está activado. Para utilizarlo correctamente, debemos tratar que nuestra primera línea de programa sea una sentencia REM seguida por, como mínimo, 18 caracteres.
De esta forma cada vez que listemos el programa, aparecerá en esa línea la fecha y hora actual en lugar de aquellos caracteres.
Lo mismo sucede cuando grabamos en disco.

```
100 CK=56328:SA=49155:PRINT"!clr!!2 cr ab!":PRINTTAB(5);
    "!rvs on!INGRESANDO DATOS, UN MOMENTO..."
110 FORI=SATOSA+151:READA:X=X+A:POKEI,A:NEXTI
120 IFX<>16817THENPRINT"ERROR EN DATAS. VERIFIQUE
    LOS VALORES":STOP
130 PRINT"!clr!":A=SA+137:POKESA+13,A-INT(A/256)*256:
```

```
    POKESA+14,INT(A/256)
140 POKESA+145,A-INT(A/256)*256:POKESA+146,INT(A/256)
150 POKESA+36,SA-INT(SA/256)*256:POKESA+37,INT(SA/256)
160 PRINT" INGRESE LA FECHA ACTUAL (MM/DD/AA)":INPUTDA$
170 IFLEN(DA$)<>8THENPRINT"DATOS INCORRECTOS.!es!
    INTENTE NUEVAMENTEQ":GOTO160
180 PRINT"!cr ab! HORA ACTUAL (HHMM)":INPUTTM$
190 IFLEN(TM$)=4ANDVAL(TM$)<1259THEN210
200 PRINT"!cr ab! ERROR EN EL INGRESO DE LA HORA.!es!
    INGRESE NUEVAMENTE!cr ab!":GOTO180
210 PRINT:INPUT" AM O PM (!rvs on!A!rvs off!/!rvs on!
    P!rvs off!)";A$:IFA$<>"A"ANDA$<>"P"THEN210
220 TM$=TM$+"00":B=0:IFA$="P"THENB=128
230 PRINT:PRINT" FECHA DEL DIA :";DA$:PRINT
240 PRINT" HORA ACTUAL    :";LEFT$(TM$,2);":";
    MID$(TM$,3,2);" ";A$;"M"
250 PRINT"!2 cr ab!":INPUT" SON LOS DATOS CORRECTOS (S/N)";A$
260 IFA$="N"THENPRINT"!clr!":GOTO160
270 FORT=1TO8:POKESA+T,ASC(MID$(DA$,T,1)):NEXT:
    IFLEFT$(TM$,2)="12"THENB=128-B
280 FORA=1TO5STEP2:D=VAL(MID$(TM$,A,1)):
    D=D*16+VAL(MID$(TM$,A+1,1))
290 IFA=1THEND=DORB
300 POKECK+3-(A-1)/2,D:NEXTA:POKECK,0
310 POKESA+142,PEEK(818):POKESA+143,PEEK(819)
320 REM PROGRAMA RELOJ
330 A=SA+15:POKE818,A-INT(A/256)*256:POKE819,INT(A/256)
340 :
```

```
350 POKESA+150,PEEK(774):POKESA+151,PEEK(775)

360 REM

370 A=SA+11:POKE774,A-INT(A/256)*256:POKE775,INT(A/256)

380 :

390 NEW

400 DATA  32, 48, 48, 47, 48, 48, 47, 48, 48, 32,  0

410 DATA   8,238,197,  3, 72,138, 72,152, 72,162,  0

420 DATA 160,  4,177, 43,201,143,208,102,200

430 DATA 177, 43,240, 97,189, 60,  3,240,  5,145

440 DATA  43,232,208,241,162,  2,177, 43,240

450 DATA  78,189,  9,220, 41,127, 74, 74, 74, 74

460 DATA   9, 48,145, 43,200,177, 43,240, 60

470 DATA 189,  9,220, 41, 15,  9, 48,145, 43,202

480 DATA 240, 12,200,177, 43,240, 43,169, 58

490 DATA 145, 43,200,208,210,200,169, 32

500 DATA 145, 43,200,173, 11,220, 48,  5,169

510 DATA  65,208,  3,234,169, 80,145, 43,200

520 DATA 169, 77,145, 43,200,177, 43,240,  6

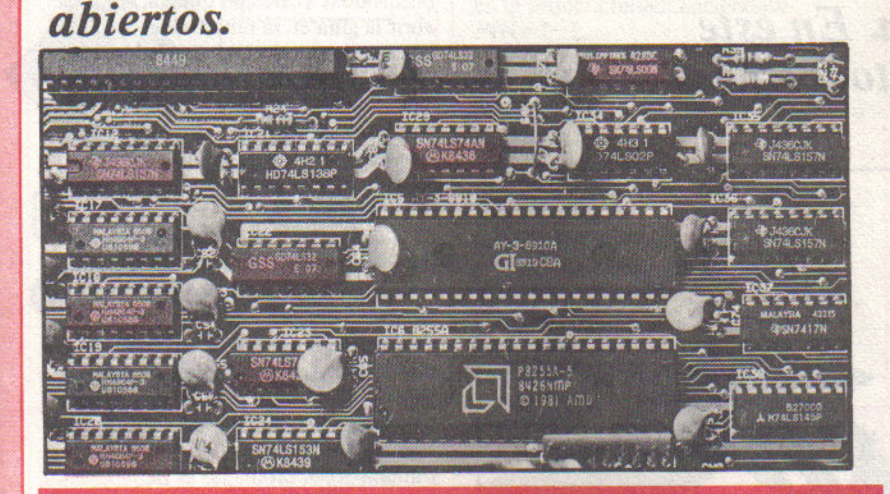
530 DATA 169, 32,145, 43,208,245,173,  8,220

540 DATA 104,168,104,170,169,  0,208,  4,104

550 DATA  76,  0,  0,206,197,  3,104, 40, 76,  0,  0
```

LAS SUBRUTINAS DEL DREAN COMMODORE 64

En este número comentamos, entre otras, la rutina que se encarga de cerrar todos los archivos que se encontraban abiertos.



Nombre de la función: CLOSE
Propósito: Cierra un archivo
Dirección de llamada: $FFC3 (hex) 65475 (decimal)
Registros de Comunicación: A
Rutina preliminar: Ninguna
Error: Código de error 0 (ver READST)
Requerimientos de stack: 2
Registros afectados: A,X,Y
Descripción: Esta rutina se utiliza para cerrar un archivo que fue abierto a través de la rutina OPEN.

Para acceder a esta rutina se debe cargar en el acumulador el número del archivo abierto y luego acceder a ella a través de uns instrucción JSR (desde el assembler) o a través de SYS (desde el basic).

Con respecto al error en la operación de esta rutina, recuerden que si al volver de ella detectamos el flag de carry en "1" (seteado), indicará que hubo error, lo que significa que el acumulador contendrá el código del mismo.

En este caso éste será el 0, lo que indica que la rutina fue suspendida por la tecla Stop.

Pasos a seguir:
1) Poner en el acumulador el número del archivo a cerrar.
2) Acceder a la rutina.

Ejemplo:
a) Desde el Assembler:
LDA #$04; cerramos archivo nro. 4
JSR $FFC3; accedemos a la rutina
b) Desde el Basic:
10 POKE780,4:REM CARGAMOS EN EL ACUMULADOR EL NUMERO DE ARCHIVO
20 SYS65475:REM ACCEDEMOS A LA RUTINA

Nota: 1) Suponemos que el archivo número 4 ya ha sido abierto.
2) Recuerden que las direcciones de memoria 780,781,782 y 783 representan al acumulador, registro X, registro Y y al registro de estado respectivamente.

Nombre de la función: CLRCHN
Propósito: Restablece todos los canales de entrada/salida.
Dirección de la llamada: $FFCC (hex) 65484 (decimal).
Registros de Comunicación: Ninguna
Rutina preliminar: Ninguna
Error: Ninguno.
Requerimientos de stack: 9
Registros afectados: A,X
Descripción: Esta rutina se utiliza para restablecer todos los canales de entrada/salida utilizados.
Además, se ejecuta automáticamente cuando se llama a la rutina CLALL.

Pasos a seguir:
Acceder a la rutina.

Ejemplo:
a) Desde el Assembler:
JSR $FFCC; accedemos a la rutina
b) Desde el Basic:
10 SYS65484: REM ACCEDEMOS A LA RUTINA

Nombre de la función: GETIN
Propósito: Toma un caracter.
Dirección de llamada: $FFE4 (hex) 65058 (decimal)
Registros de Comunicación: A
Rutina preliminar: CHKIN, OPEN
Error: Ver READST
Requerimientos de stack: 7
Registros afectados: A,X,Y
Descripción: Esta rutina se utiliza para tomar un caracter desde el canal actualmente abierto.

Para ello se debe especificar el canal a través de las rutinas CHKIN y OPEN. Si no se especifica ese canal, el sistema operativo asume por default que se trata del teclado.
En este caso se toma el caracter desde la cola del teclado y se lo deposita en el acumulador (su código ASCII).
El usuario se desentiende de poner los caracteres en esa cola ya que 60 veces por segundo se ejecuta una rutina (denominada SCNKEY) que barre el teclado y pone los caracteres oprimidos en dicha cola, cuya longitud es de 10 caracteres.
En caso de que se opriman más de diez caracteres, los sobrantes se perderán.

Pasos a seguir:
1) Acceder a la rutina.
2) El código ASCII del caracter leído es devuelto en el acumulador. Si no se oprimió ninguna éste tendrá cero (0).

Ejemplo:
a) Desde el Assembler:
INICIO
JSR $FFE4; accedemos a la rutina
CMP #$00; vemos si se oprimió una tecla.
BEQ INICIO; si no volvemos
JSR $FFD2; lo imprimimos.
b) Desde el Basic:
10 SYS65058
20 IFPEEK(780)=0THEN10:REM VOLVEMOS SI NO SE OPRIMIO NINGUNA TECLA
30 SYS65490: REM LO IMPRIMIMOS EN PANTALLA

BUSQUEDA DE DATOS

En el número anterior hemos explicado algunos métodos para ordenar información. En éste comentamos dos métodos para poder hallar datos.



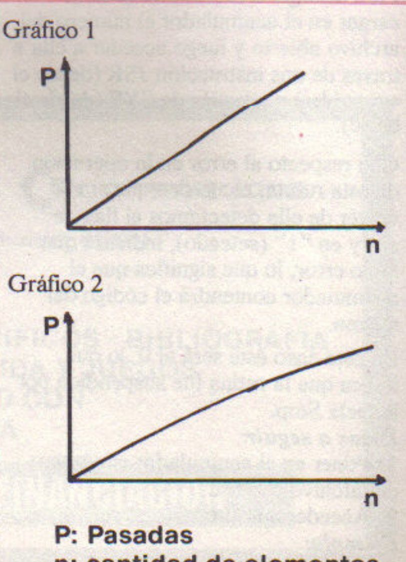
Generalmente surge la necesidad de saber si un determinado ítem se encuentra almacenado en algún tipo de estructura de datos, como lo son por ejemplo los vectores y matrices.
Todo programa utilitario que requiera tomar información desde algún periférico, emplea algún método de búsqueda para hallarlo.
Nosotros seguiremos utilizando los vectores para explicar los diversos métodos.
Básicamente existen dos algoritmos. Cada uno requiere que el bloque de datos donde se buscará uno en particular esté "preparado". Ellos son:

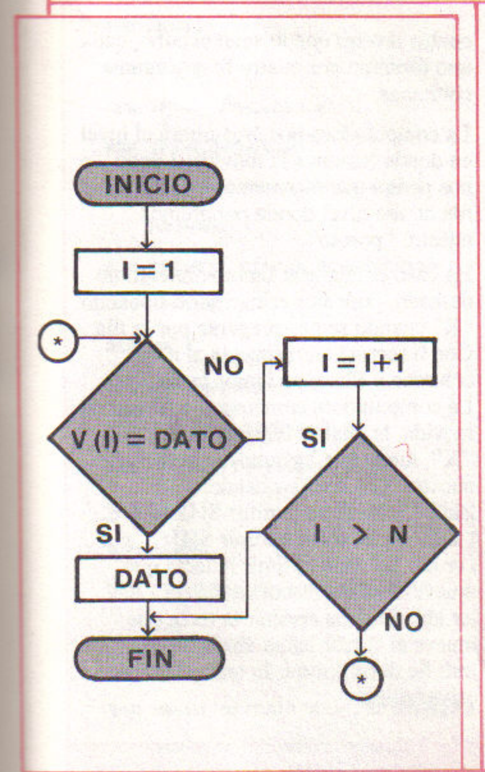
1) Búsqueda secuencial

El principio de funcionamiento del algoritmo es muy sencillo: se va buscando el dato a través del bloque en forma secuencial.
Una vez hallado se suspende la búsqueda.
Por como realiza la tarea no se requiere que el bloque esté ordenado.
La desventaja del método es que, si ese bloque contiene demasiados datos, el tiempo de búsqueda puede ser muy alto.
Imaginen un vector con más de 1000 elementos y que el dato buscado se encuentre en la posición 1000.
Antes de hallarlo debemos pasar por los 999 anteriores.
Un ejemplo claro es el de buscar, usando este método, el apellido Torres en la guía telefónica. Debemos pasar primero por la A, B, C, ..., hasta llegar a la T.
Este método es conveniente utilizarlo cuando la cantidad total de datos es chica, digamos a lo sumo 10 elementos.
En caso de que el dato buscado sea el primero, el problema queda resuelto.
En cambio, si el elemento buscado es el que se encuentra en la posición n, debemos realizar n-1 pasadas.
En general el número promedio de pasadas es de $(1+n)/2$.
El gráfico 1 representa esta ecuación. Observen que a medida que aumenta la cantidad de elementos aumenta el número de pasadas.
El listado 1 representa el diagrama de flujo y el programa correspondiente a este método.

2) Búsqueda binaria

El segundo método, y tal vez el más eficaz hasta el momento, necesita que los datos estén ordenados. De otra manera no puede ser utilizado.
El principio de funcionamiento consiste en ir dividiendo sucesivamente el bloque de datos hasta encontrar el buscado.
Siguiendo con el ejemplo de la guía, buscando a Torres, lo que hacemos es abrir la guía en la mitad. Si el apellido que se encuentra en el encabezamiento de la página es alfabéticamente menor que el buscado, entonces éste se encuentra dentro de la segunda mitad.
Ahora tomamos esa mitad y la volvemos a dividir en dos, repitiendo el procedimiento.
Si el dato a buscar se encuentra en la mitad del bloque el problema quedaría resuelto, ya que lo hemos encontrado en la primera pasada.
En cambio, y sin entrar en detalles matemáticos, si el elemento se hallaba en la posición n, el número de pasadas requeridas para encontrarlo hubiese sido de $\log_2 n$.
Finalmente el promedio es de $(1+\log_2 n)/2$. El gráfico 2 representa esta ecuación. Observen que a medida que aumentan los elementos, el número de pasadas crece logarítmicamente.
El listado 2 representa el diagrama de flujo y el programa de este método.
Primero seteamos las variables TOPE y FIN a 1 y n+1 (donde n representa el total de elementos). Seguidamente preguntamos si TOPE es igual a FIN.
Si la igualdad es cierta, indicará que el elemento no se halla en el bloque de datos.
Si no es cierta procedemos a calcular el promedio de ambos.

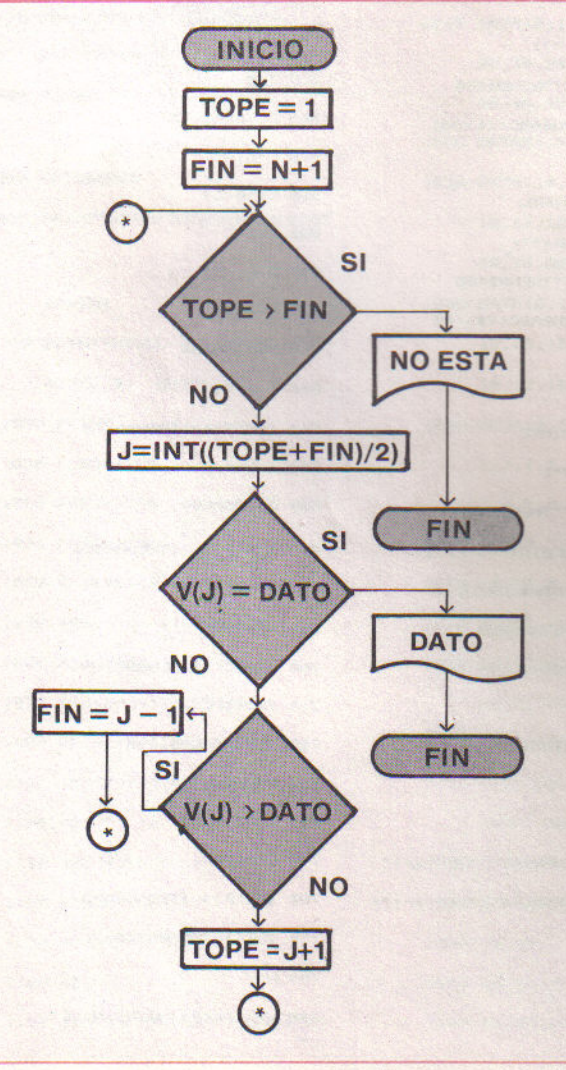




Vemos si el elemento ubicado en la posición J es el dato buscado.
Si lo es imprimimos el mensaje correspondiente. Si no, analizamos en qué mitad puede encontrarse. En caso de que el elemento buscado sea mayor que el que apunta J, se parte de la suposición que éste se encuentra en la segunda mitad, haciéndose TOPE=J+1 y volviendo a partir este nuevo intervalo en dos.
Si el elemento buscado es menor que el que apunta J, el dato puede ser hallado en la primera mitad, haciéndose FIN=J−1.
Cada vez que cambiamos los punteros de FIN y TOPE, repetimos el procedimiento tomando como un nuevo bloque de datos a alguna de las dos mitades actuales.
Recuerden que este método necesita que los datos estén ordenados.
Tanto para clasificación como para búsqueda de datos, serían muy útiles sentencias como WHILE o REPEAT, ambas orientadas a la programación estructurada, que no se encuentran incluídas en el Basic de la C-64.

```
Listado 1

10 DIMV(100)
20 FORI=1TO100
30 V(I)=I
40 NEXTI
50 INPUT"INGRESE DATO";DAT0
60 FORI=1TO100
70 IFV(I)=DAT0THENPRINT"EL
   DATO SE ENCUENTRA":STOP
80 NEXTI
90 PRINT"EL DATO NO
SE ENCUENTRA"
```

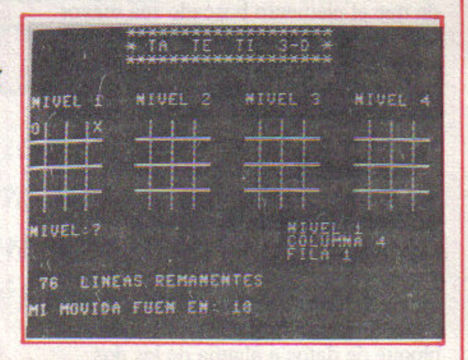



```
Listado 2

10 DIMV(101)
20 FORI=1TO101
30 V(I)=I
40 NEXTI
50 INPUT" INGRESE DATO";DAT0
60 TPE=1:FI=101
70 IFTPE=FIANDV(FI)<>DAT0THEN
   PRINT"EL DATO NO SE ENCUENTRA"
   :STOP
80 J=INT((FI+TPE)/2):PRINTJ;
90 IFV(J)=DAT0THEN PRINT"EL
   DATO SE ECUENTRA":STOP
100 IFV(J)>DAT0THENFI=J-1:GOTO70
110 TPE=J+1:GOTO70
```

TA TE TI 3-D

¿Quién no ha jugado alguna vez a este tradicional juego? ¿Pero, han tratado de practicarlo en tres dimensiones?



A diferencia del típico TA TE TI, el juego que aquí presentamos tiene los mismos objetivos que el anterior, pero se juega sobre un cubo de 4 x 4 x 4, es decir es un TA TE TI en tres dimensiones.

Así el ta te ti se podrá formar usando las diagonales (en profundidad), las filas o las columnas.
Primeramente se nos pregunta si necesitamos instrucciones.
Luego aparecerá sobre la pantalla los cuatro niveles que forman el cubo, cada uno formado por cuatro filas y cuatro columnas.

La computadora nos preguntará el nivel en donde haremos la movida. Luego nos pedirá que ingresemos la columna y fila de ese nivel donde pondremos nuestro "poroto".

En caso de ingresar un movimiento no deseado, podemos reingresarlo tipeando "R" cuando se nos pregunte por la fila. Con 0 (ceros) en respuesta al nivel, columna y fila reiniciamos la partida.
La computadora comenzará a pensar su movida, la cual se representa con una "X", luego que hayamos ingresado la nuestra. Los códigos usados son:
¡shift C!: Se debe oprimir SHIFT y C.
¡ shift +!: Se debe oprimir SHIFT y +.
¡ cr ar!: Se debe oprimir la tecla que mueve el cursor hacia arriba.
¡cr ab!: Se debe oprimir la tecla que mueve el cursor hacia abajo.
¡es!: Se debe oprimir la tecla espaciadora.

```
100 B$="********************"
110 A$="!35 es!
120 PRINT"!clr!"TAB(3)B$:PRINT"
    * TA  TE  TI  3-D *":PRINT
    TAB(9)B$
130 IFF=1THEN00180
140 PRINT"!2 cr ab!NECESITA
    INSTRUCCIONES  ?(S/N)"
150 GETY$:IFY$=""THEN00150
160 IFY$="S"THEN01420
170 PRINT"!cr ar!"A$"!2 cr ar!";
180 RESTORE:F=0
190 DIMA(75),B(75)
200 DIMG(63),V(63)
210 DIMN0(3),N1(3)
220 FORI=0TO75:READA(I):NEXTI
230 FORI=0TO75:READB(I):NEXTI
240 FORI=0TO3:READN0(I):NEXTI
250 FORI=0TO3:READN1(I):NEXTI
260 FORI=0TO63
270 G(I)=0
280 NEXTI
290 FORI=0TO63
300 V(I)=0
310 IFG(I)=0THEN00330
320 V(I)=-1
330 NEXTI
340 GOSUB00980
350 PRINT"NIVEL:?";
360 GETY$:IFY$=""THEN00360
370 IFASC(Y$)>52ORASC(Y$)<48
    THEN00360
380 L=VAL(Y$):PRINTL
390 PRINT"COLUMNA:?";
400 GETY$:IFY$=""THEN00400
410 IFASC(Y$)>52ORASC(Y$)<48
    THEN00400

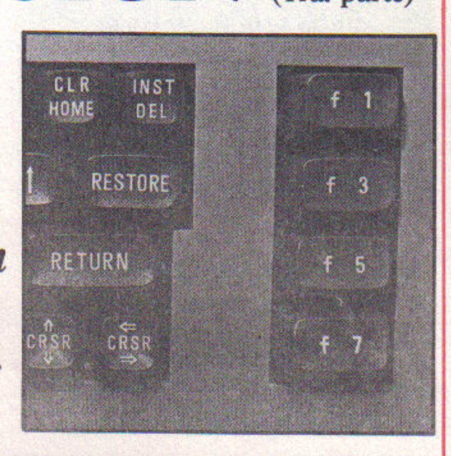
420 C=VAL(Y$):PRINTC
430 PRINT"FILA:?";
440 GETY$:IFY$=""THEN00440
450 IFY$="R"THENPRINT"!3 cr ar!"
    :PRINTA$:PRINTA$:PRINTA$"
    !3 cr ar!":GOTO00350
460 IFASC(Y$)>52ORASC(Y$)<48
    THEN00440
470 R=VAL(Y$):PRINTR
480 P=16*R+4*L+C-21
490 IFP=-21THENRUN
500 IFV(P)=-1THENPRINT"!3 cr ar!"
    A$:PRINTA$:PRINTA$"!3 cr ar!":
    GOTO00350
510 G(P)=-1
520 V(P)=-1
530 GOSUB00980
540 N=-1
550 T=0
560 FORL=0TO75
570 S=A(L)
580 V1=B(L)
590 C=0
600 H=0
610 P=S-V1
620 FORI=0TO3
630 P=P+V1
640 IFG(P)<>1THEN00670
650 C=C+1
660 GOTO00690
670 IFG(P)<>-1THEN00690
680 H=H+1
690 NEXTI
700 IFH*C<>0THEN00810
710 T=T+1
720 IFH=4THEN01290
730 P=S-V1
740 FORI=0TO3
750 P=P+V1
760 IFV(P)=-1THEN00800
770 IFC<>3THEN00790
780 N=P
790 V(P)=V(P)+N0(H)+N1(C)
```

```
800 NEXTI
810 NEXTL
820 IFN<>-1THEN01310
830 IFT=0THEN01360
840 PRINTA$:PRINTA$:PRINTA$:
    PRINTA$
850 PRINTT" LINEAS REMANENTES"
    :PRINT
860 M=-1
870 FORI=0TO63
880 IFM>=V(I)THEN00910
890 M=V(I)+INT(RND(TI)*5)
900 P=I
910 NEXTI
920 PRINT"MI MOVIDA FUEN EN:"
    ;M;
930 G(P)=1
940 PRINTTAB(24)"!6 cr ar!NIVEL";
    INT(P/4)-4*INT(P/16)+1
950 PRINTTAB(24)"COLUMNA";
    P-4*INT(P/4)+1
960 PRINTTAB(24)"FILA";
    INT(P/16)+1
970 GOTO00290
980 PRINT"S":PRINT:PRINT:PRINT

    :PRINT
990 PRINT"NIVEL 1   NIVEL 2
    NIVEL 3   NIVEL 4"
1000 PRINT
1010 FORR=1TO4
1020 FORL=1TO4
1030 FORC=1TO4
1040 P=16*R+4*L+C-21
1050 IFG(P)<>-1THEN01070
1060 PRINT"O";
1070 IFG(P)<>1THEN01090
1080 PRINT"X";
1090 IFG(P)<>0THEN01110
1100 PRINT" ";
1110 IFC<4THENPRINT"|";
1120 NEXTC
1130 PRINT"   ";
1140 NEXTL
1150 IFR<4THENPRINT"!shift C!
    !shift +!!shift C!!shift +!
    !shift C!!shift +!!shift C!
    !3 es!!shift C!!shift +!
    !shift C!!shift +!!sh4ft C!
    !shift +!!shift C!!3 es!
    !shift C!!shift +!!shift C!
    !shift +!!shift C!!shift +!
    !shift C!!3 es!!shift C!
    !shift +!!shift C!!shift +!
    !shift C!!shift +!!shift C!
1160 NEXTR
1170 PRINT
1180 RETURN
1190 DATA0,0,0,1,2,3,3,4,8,12,
    0,0,1,2,3,3,0,0,1,2,3,3,4,
    4,5,6,7,7,8,8
1200 DATA9,10,11,11,12,12,13,
    14,15,15,12,12,13,14,15,15,
    16,16,16,17,18
1210 DATA19,19,20,24,28,32,32,
    32,33,34,35,35,36,40,44,48,
    48,48,49,50
1220 DATA51,51,52,56,60
1230 DATA1,4,5,4,4,3,4,1,1,1,
    16,17,16,16,15,16,20,21,20,
    20,19,20,16,17
1240 DATA16,16,15,16,16,17,16,
    16,15,16,12,13,12,12,11,12,
    16,17,16,16
1250 DATA15,16,1,4,5,4,4,3,4,
    1,1,1,1,4,5,4,4,3,4,1,1,1,
    1,4,5,4,4,3,4
1260 DATA1,1,1
1270 DATA0,3,15,35
1280 DATA1,4,9,40
1290 PRINTA$:PRINTA$:PRINTA$:
    PRINT"UD GANO"
1300 GOTO01370
1310 PRINTA$:PRINTA$:PRINTA$:
    PRINT"YO GANE"
1320 G(N)=1
1330 GOSUB00980
1340 PRINT
1350 GOTO01370
1360 PRINTA$:PRINTA$:PRINTA$:
    PRINT"JUEGO EMPATADO"
1370 PRINT"OTRO JUEGO? (S/N)"
1380 GETY$:IFY$=""THEN01380
1390 IFY$="S"THENPRINT"qq"A$:
    PRINTA$:PRINTA$:PRINTA$:
    PRINTA$:PRINTA$:GOTO00260
1400 PRINT"!home!":PRINT:PRINT:
    PRINT:PRINT:PRINT:PRINT
    TAB(10)"FIN"
1410 END
1420 F=1:PRINT"!cr ar!EL JUEGO SE
    DESARROLLA SOBRE UN CUBO DE
    4 X 4 X 4"
1430 PRINT"EL OBJETIVO ES
    HACER TA TE TI SOBRE EL,"
1440 PRINT"EN ALGUNA DIRECCION
    (INCLUYENDO DIAGONALES."
1450 PRINT"LA COMPUTADORA
    CONTESTARA CON UNA 'X'."
1460 PRINT"UD JUEGA PRIMERO.
    SERA INTERROGADO POR"
1470 PRINT"EL NIVEL, LA
    COLUMNA Y FILA"
1480 PRINT"DONDE UD QUIERE
    JUGAR. LOS NIVELES ESTAN"
1490 PRINT" ENUMERADOS DE 1 A
    4, LAS COLUMNAS DE 1 A 4"
1500 PRINT"Y LAS FILAS DE 1 A
    4"
1510 PRINT"SI INGRESA 0 EN
    RESPUESTA AL NIVEL, COLUMNA Y
    FILA"
1520 PRINT"EL JUEGO COMENZARA
    NUEVAMENTE"
1530 PRINT"SI INGRESA UN
    MOVIMIENTO QUE UD CONSIDERA
    ERRONEO"
1540 PRINT"INGRESE 'R' EN
    RESPUESTA A FILA?"
1550 PRINT"ASI PODRA
    REINGRESAR LA MOVIDAD DESEADA."
1560 PRINT:PRINT"OPRIMA
    ALGUNA TECLA"
1570 GETY$:IFY$=""THEN01570
1580 GOTO00120
```

COMO DEFINIR LAS TECLAS DE FUNCION (1ra. parte)

En esta nota explicamos cómo se pueden definir las teclas de función para que impriman comandos, instrucciones y sentencias.



Algunos intérpretes Basic, como el Simon's Basic, permiten al usuario poder definir las teclas de función a través de comandos determinados, como lo es el KEY.
De esta manera podemos ejecutar, por ejemplo, el comando LIST con sólo oprimir la tecla de función uno (F1), o ejecutar RUN a través de F2, etc.
Nosotros podemos realizar esta tarea sin necesidad de recurrir al Simon's. Basta con conocer cómo es el procedimiento que lo permite hacer.

I-ERRE-CU

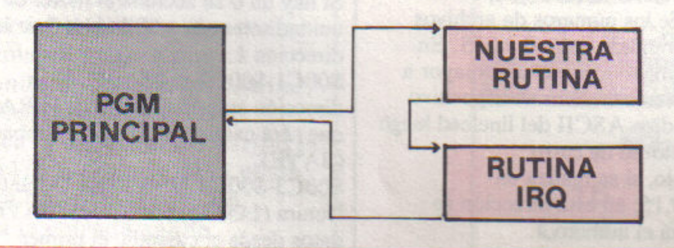
Existe una rutina denominada IRQ, una de las más importantes del sistema, que 60 veces por segundo realiza, entre otras cosas, el barrido del teclado.
En caso de que se haya oprimido alguna tecla, ésta la deposita en el buffer del teclado, el cual tiene una longitud de diez caracteres.
Este buffer se encuentra ubicado a partir de la dirección 631 y se extiende hasta la dirección 640 (direcciones hexadecimales $0277-$0280).
Es el que nos permite teclear caracteres mientras ejecutamos un comando de disco. En caso de superar los diez caracteres los restantes se perderán debido a que se encuentra lleno.
Como la mayoría de las rutinas del Drean Commodore 64, la rutina IRQ tiene un puntero en la RAM que le indica al sistema operativo dónde se encuentra, es decir su ubicación en la memoria ROM.
Si nosotros modificamos esos punteros para que señalen a otra dirección donde se encuentra nuestra rutina, la estaremos ejecutando periódicamente cada 1/60 segundos.
Claro que, luego de finalizarla, debemos ejecutar la rutina IRQ normalmente para que siga realizando las tareas antes descriptas.
De otra manera se quedará en un lazo o loop infinito, por lo que tendremos que resetear la computadora para normalizarla (la tecla STOP no podrá ser detectada).
La figura 1 muestra como debemos insertar nuestra rutina.
Los punteros que antes mencionamos se encuadran en las direcciones $0314 y $0315 (788 y 789). El contenido de estos es, a menos que se hallan modificado, $31 y $EA, respectivamente.
De acuerdo al formato de almacenamiento de direcciones de la C-64, estos indican que la rutina se encuentra en la dirección $EA31 (ROM).
Nosotros debemos modificarlos para que apunten a nuestra rutina, la cual se encargará de detectar si se oprimió alguna de las teclas de función y, en base a ello, ejecutar un comando predefinido.
Pero antes hagamos un pequeño programa que vaya cambiando el color del cursor, obviamente 60 veces por segundo.
Los pasos principales son:
1) Modificar los punteros para que señalen a nuestra rutina.
2) Incrementar el contenido de la dirección de color del cursor.
3) Saltar a la rutina IRQ para que realice las tareas normales.
El paso 1 se realiza sin demasiados problemas. Antes de modificarlos se le debe decir al sistema que no realice interrupción. Esto se hace a través de la instrucción SEI (SEt Interrupt).
Esta pone en "1" el bit de interrupción del registro de estado prohibiendo la IRQ.
Esto se hace para que no se produzca una interrupción en el momento en que se está cambiando los punteros.
Imaginen lo que pasaría si, luego de modificar la parte baja de la dirección, se produce una interrupción. El control se transferirá a un área indeterminada, produciéndose un posible "cuelgue" de la C-64.
El programa que realiza el cambio de color del cursor corresponde al listado 1. El listado 2 es su equivalente en Assembler.
Al ejecutar el programa del listado 1, verán como va cambiando el color del cursor. Pueden restablecer el sistema oprimiendo STOP y RESTORE al mismo tiempo.
Observen que la rutina que se encarga de cambiar el color del cursor es una sola instrucción.
Sólo incrementamos el contenido de la dirección $0287 (647) que le indica al sistema el código de color del cursor.
Probemos de hacer POKE647,CO donde CO es un valor comprendido entre 0 y 16, en modo directo.
Para nuestro objetivo principal (definir las teclas de función) debemos chequear qué tecla de función se oprimió y, en base a ello, ejecutar un comando o sentencia.
Este chequeo lo debemos poner antes de que se ejecute la rutina IRQ, es decir como se explica en el gráfico 1.
Para ello utilizaremos tres cosas:
a) La dirección $C5, que contiene el código de la matriz de la tecla que se oprimió.
b) La dirección $C6, que indica la cantidad de caracteres que hay en el buffer del teclado.
c) El buffer del teclado.
Para entender mejor lo que indica el punto carguen y ejecuten el listado 3. Observen los valores que se imprimen cuando se oprime una tecla.
Presten más atención a los valores que aparecen cuando se oprimen F1, F3, F5 y F7. Comprueben que estos son 4, 5, 6 y 3, respectivamente.

Listado 1

```
10 FORI=49152TO49170:READA:POKEI,A
20 C=C+A:NEXT:IFC<>1909THENPRINT"ERROR EN DATAS":STOP
30 SYS49152:NEW
100 DATA 120,169, 13,141, 20,  3
110 DATA 169,192,141, 21,  3, 88,96
120 DATA 238,134,  2, 76, 49, 234
```

Listado 2

```
C000 SEI          ;suspendemos interrup
C001 LDA #$0D     ;modificamos punteros
C003 STA $0314    ;para que apunten a
C006 LDA #$C0     ;nuestra rutina que
C008 STA $0315    ;comienza en $C00D
C00B CLI          ;normalizamos interrup
C00C RTS          ;regresamos al Basic
C00D INC $0287    ;incrementamos codigo de color bajo cursor
C010 JMP $EA31    ;ejecutamos IRQ normalmente
```

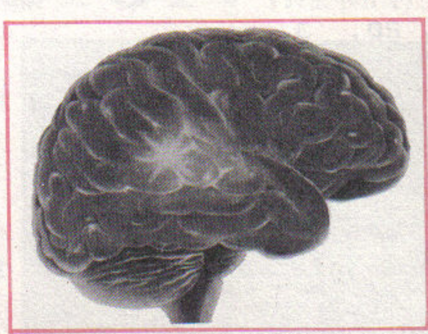


Listado 3

```
10 PRINTPEEK(197)
20 GOTO10
```

MAPA DE MEMORIA

Continuamos describiendo las direcciones que constituyen el rango $0090-$00FF.



$00AA (170): Esta dirección es usada por la rutina de cinta y por la RS-232. La primera la utiliza para saber si el caracter recibido es un dato o si es sincronismo.
$00AB (171): Esta dirección se utiliza para saber si un dato se perdió durante la transmisión a través de la RS-232.
$000AC-$00AD (172-173): Aquí se copia el puntero que indica el área de memoria RAM que será cargada (LOAD) o grabada (SAVE).
También es temporalmente utilizada por la rutina que administra la pantalla.
$00AE-$00AF (174-175): El contenido de esta dirección es seteada por la rutina SAVE, poniendo aquí la dirección de fin del texto basic, la cual es utilizada por los comandos LOAD, SAVE y VERIFY.
$00B0-$00B1 (176-177): Estas direcciones se utilizan durante la lectura de cinta.
$00B2-$00B3 (178-179): Esta dirección se utiliza para indicar el inicio del buffer de cassette. Comunmente ésta comienza a partir de la dirección $033C (828).
$00B4 (180): La rutina que utiliza la RS-232 usa esta dirección para contar los bits trasmitidos y para manejar el bit de stop y de paridad.
$00B5 (181): Aquí se almacena el próximo bit a transmitir por la RS-232 y, además, para indicar cuando se produce fin de cinta (EOT).
$00B6 (182): Esta dirección se utiliza durante la transmisión de datos a través de la RS-232.
$00B7 (183): En esta dirección se almacena el número de caracteres que tiene el actual nombre de un archivo (incluyendo nombres de programas). La longitud de ellos es de 16 caracteres (en caso de tratarse de programas referentes a la unidad de disco).
Para cinta ésta puede ser de hasta 187 caracteres.
Si se utiliza la unidad de disco, esta dirección siempre tendrá un número mayor que cero. Como mínimo puede tener un uno en caso que se ejecute un LOAD"*",8.
Teniendo la unidad de cinta, aquí sí puede haber un cero, lo cual correspondería al comando LOAD.
$00B8 (184): Esta dirección contiene el número de archivo actual.
Un máximo de 10 archivos pueden estar abiertos al mismo tiempo.
El rango de los números de archivos está comprendido entre 0 y 255. En caso de utilizarse un número mayor a 127, el sistema enviará al dispositivo serie el código ASCII del linefeed luego de cada retorno de carro.
Por ejemplo, si se realiza un OPEN4,8,15, en esta dirección se almacenará el número 4.
$00B9 (185): Esta dirección contiene el número de dirección secundaria, es decir, el comando que se le enviará a un periférico.
En caso de ser serie, el rango de valores permitido está comprendido entre 0 y 31, y desde 0 hasta 127 para otros dispositivos.
Como mencionamos, la dirección secundaria es un comando que se le envía al periférico asociado.
Por ejemplo, abrir un archivo a través de OPEN 4,8,15, (aquí la dirección secundaria es 15), permite que el usuario se comunique directamente con el DOS.
Cuando realizamos la carga de un programa a través de, por ejemplo, LOAD"TEST",8,0 (aquí la dirección secundaria es 0), el programa comenzará a cargarse a partir de la dirección que indica el puntero de inicio de texto basic.
A través de LOAD"TEST",8,1 el programa comenzará a cargarse a partir de la dirección indicada en el mismo programa (recuerden que los programas almacenados en el disco tienen dos sectores reservados para indicar la dirección, en formato bajo alto, de la dirección de inicio).
$00BA (186): En esta dirección se almacena el número de dispositivo actual. Este puede ser 0 (teclado), 1 (datassette), 2 (RS-232), 3 (pantalla), 4-5 (impresora), 8-11 (disco).
$00BB-$00BC (187-188): Los contenidos de estas direcciones contienen la dirección (formato parte alto-bajo) del archivo actual.
$00BD (189): Esta dirección es utilizada por la rutina RS-232 como un byte de paridad.
También la utiliza la rutina que lee un programa de cinta para almacenar el caracter que será leído o transmitido.
$00BE (190): Esta dirección la utiliza la rutina de cinta para saber cuantos bytes le quedan para leer o para escribir.
$00BF (191): Esta dirección también es utilizada por la rutina de cinta.
$00C0 (192): Cuando se oprime la tecla de operación de la unidad de cinta, la rutina IRQ mira en esta dirección para ver qué valor hay almacenado.
Si hay un 0 se acciona el motor de la unidad seteando a "0" el bit 5 de la dirección 1.
$00C1-$00C2 (193-194): Esta dirección apunta al inicio de la RAM que será cargada (LOAD) o grabada (SAVE).
$00C3-$00C4 (195-196): Durante la lectura (LOAD) o escritura (SAVE) de datos desde el cassette, el primer bloque leído, el cual contiene el encabezamiento respectivo, es cargado desde o hacia el buffer del cassette.
El resto de estas direcciones es indicar el inicio, en esa RAM, en donde se cargarán esos datos.
$00C5 (197): Esta dirección es continuamente seteada, por la rutina IRQ, con el código de la última tecla oprimida.
El valor que aquí se ingresa es el que proviene de la matriz del teclado, es decir que no están codificados por sus valores en ASCII.
Para visualizar esto último ejecuten este programa:

```
10 PRINT PEEK (197)
20 GOTO 10
```

Reseteando la C-64

A continuación describiremos las distintas maneras de resetear la Drean Commodore 64 por software. La herramienta que se utiliza es una sola: el comando SYS. Las formas son:
SYS 64759: Actúa de la misma manera que si presionamos las teclas RUN/RESTORE simultáneamente.
Este resetea el chip de sonido y color, limpia la pantalla e imprime el "READY". Los programas en memoria no son borrados.
SYS 64738: Simula el proceso de activación de la C-64. Restablece los punteros de comienzo y fin de programa basic. Los programas no son borrados con lo que se los puede recuperar tipeando POKE2050,1:SYS42291. Si se borran los datos que se encontraban en el buffer del cassette (direcciones 828-1019). Si se han desarrollado programas en lenguaje máquina escribiéndose a partir de la dirección 49152 (RAM alternativa), éstos tampoco se borran si se acciona este SYS.
SYS 64760: No destruye el área del buffer del cassette. Tampoco borra la memoria principal pero sí resetea los punteros. Los programas se pueden recuperar de la misma manera que en el caso anterior. Se reinicializa las entradas/salidas pero no se chequea la conexión del cartridge.
SYS 64763: Trabaja de la misma manera que 64760 pero no resetea la entrada-salida.

Directorio selectivo

Para los que tengan la disketera 1541, aquí les suministramos algunos comandos para cargar programas y archivos específicos:
LOAD"$*=P",8: Carga archivos tipo PRG.
LOAD"$*=S",8: Carga archivos tipo SEQ.
LOAD"$*=R",8: Carga archivos tipo REL.
LOAD"$*=U,8": Carga archivos tipo USR.

Pantalla inversa

Con el siguiente programa podrán poner la pantalla en video inverso con sólo tipear SYS52992. Cuando ejecuten el programa por primera vez se chequeará los DATA del programa a través de un CHECKSUM. En caso de que los valores sean incorrectos, el programa informará de ello y suspenderá la ejecución. Caso contrario nos pedirá que borremos las líneas 1, 2, 3, 4, 5 y 6, que efectuemos RUN y, finalmente, SYS52992. A continuación la pantalla se pondrá en video inverso.

```
1 REM VIDEO INVERSO
2 FORY=1TO27:READZ:CS=
CS+Z:NEXT
3 IFCS    4345THENPRINT
"ERROR EN DATAS":STOP
4 PRINT"BORRA LAS 1,2,3,4,
5 y 6"
5 PRINT"DARLE RUN Y
LUEGO"
6 PRINT "TIPEA SYS52992":
END
10 FORJ=52992TO53018:
READK:POKEJ,K:NEXT
20 DATA169, 0, 133, 251, 169, 4,
133, 252
30 DATA162, 4, 160, 0, 177, 251,
73, 128
40 DATA145, 251, 200, 208, 247,
230, 252, 202
50 DATA208, 240, 96
```

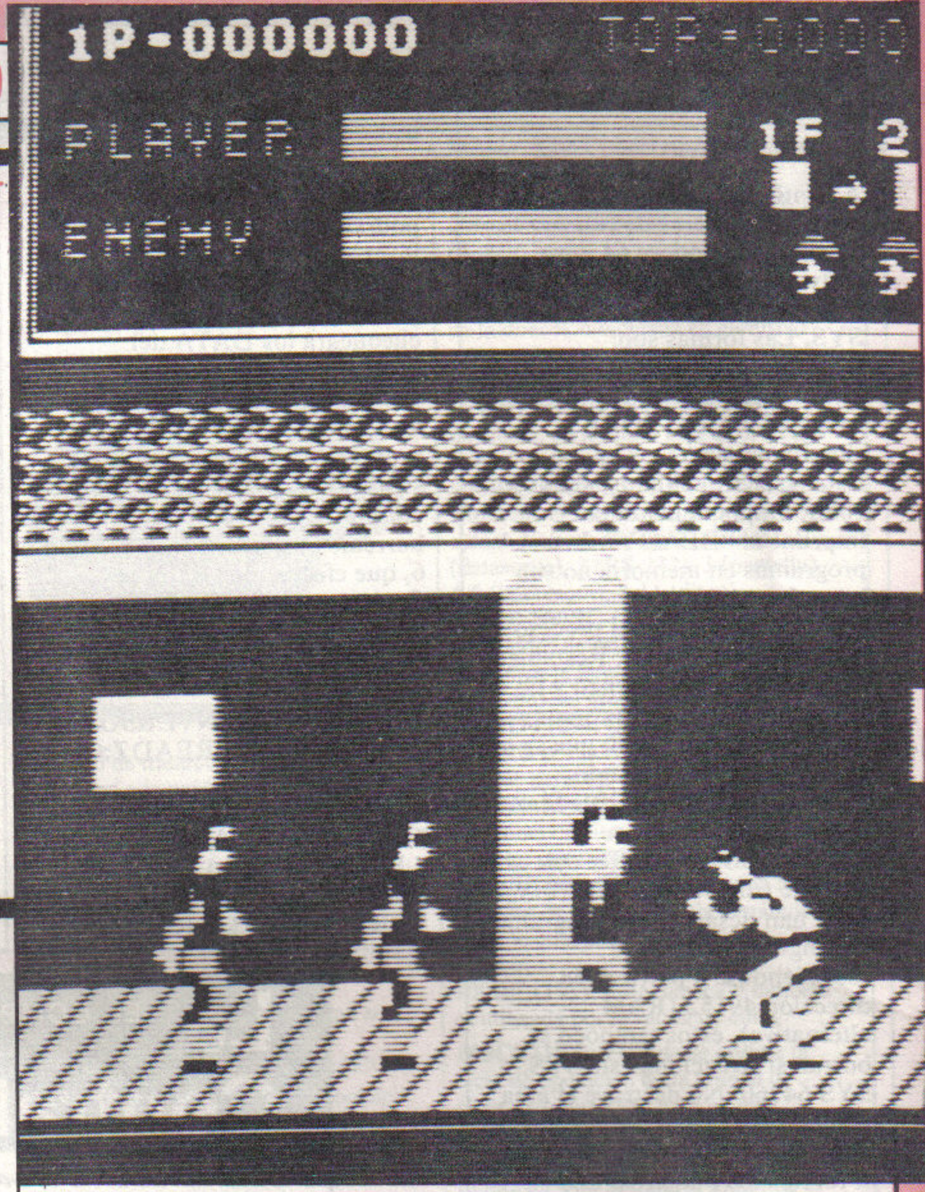
Océano en la C-64

Si ejecutamos este programa oiremos como las "olas" llegan a la costa.

```
20 V=54272:POKEV+6,240:
POKEV+4,129:POKEV+1,34:
POKEV,75
30 FORA=1TO15:POKEV+24,
A:FORD=1TO50:NEXTD:
NEXTA
40 FORB=15TO1STEP−1:
POKEV+24,B:FORD=1TO200:
NEXTD:NEXTB
50 FORD=1TO600+1200*
RND(0):NEXTD:GOTO20
```

KUNG-FU

Rating total: B
Creatividad: A
Documentación: B
Profundidad: del juego: A—
Valor en relación al precio: Se justifica
Mantiene el interés: Si
Computadora: Drean Commodore 64
Editor: ?



Si se quieren convertir en "maestros" del arte marcial o en cinturones negros, sólo deben jugar con este juego y atravesar cinco pisos llenos de feroces Karatecas.
El participante o los participantes pueden seleccionar a partir de qué piso comenzarán a jugar. Cuando mayor sea éste, mayor será la dificultad del combate.
El objetivo del juego es guiar al príncipe DONKE ESKA, maestro en Kung-Fu, a través de los cinco pisos y rescatar a la princesa MIKA KE MOKO.
Cada uno de ellos tiene distintos tipos de combate. Por ejemplo, en el primer piso, sólo debemos enfrentarnos con simples Karatecas.
Sin embargo, en cada piso, siempre aparece un lanza cuchillos que no duda en lanzarnos estos lastimosos objetos. Para contrarrestarlo podemos saltar o agacharnos, para luego responder con nuestra dura represalia.
Nuestro príncipe puede saltar, agacharse, pegar patadas y trompadas. Esto último se hace conmutando el tipo de golpe a través de la barra espaciadora.
Si somos tomados por los secuaces del malvado rey KATATE PIKO, tenemos que liberarnos moviendo rápidamente el joystick de un lado hacia el otro.
De otra manera nuestra vida irá bajando hasta llegar a cero, en donde estaremos más que muertos. Sólo tenemos tres vidas.

Si somos tan hábiles de superar los 4000 puntos, se nos dará una vida más.
En cada piso debemos enfrentar a un especialista en arte marcial y secuaz del rey KATATE PIKO. En el primer piso debemos enfrentarnos a un "artesano" del TAE KUONDO.
El, a través del típico "palito", tratará de disminuir nuestra vida o mejor dicho nuestro nivel de vida.
Nosotros a través de "patadas" y "piñas" tenemos que aniquilarlo a él.
En el segundo piso nos debemos enfrentar a un lanza "estrellitas". Claro que estas estrellitas no son las que se prenden para Navidad. Estas duelen.
Además, a medida que nos acercamos a la escalera que nos lleva al tercer piso, aparecen unos enanitos que, si no saltamos o no les pegamos, nos toman de los tobillos inmovilizándonos.
También debemos esquivar jarrones que caen desde el techo y que, al estrellarse contra el piso, liberan unas "cucarachas" venenosas.
En el tercer piso no hay especialista, ¡sólo un gigante de unos dos metros y medio!!
Si nos llega a pegar, hemos calculado que nuestro cuerpo llega a recorrer, en el aire, hasta 3 metros.
En el cuarto piso se encuentra un mago. Tiene poderes para hipnotizarnos y para "duplicarse". Debemos tener mucho cuidado si no queremos perecer por acción de algún polvo mágico.
Además, durante el recorrido, tenemos que esquivar a unas abejas que tratarán de picarnos. Cada una de ellas vale, en caso de matarlas, 500 puntos.
En el quinto piso, el último, el especialista que se encuentra aquí nos hace recordar a KUAI-CHAI KEIN, el ídolo de aquella famosa serie llamada KUNG-FU.
Es capaz de tirar "patadas", "piñas", saltar y agacharse. Todo ésto lo hace para esquivar nuestro ataque.
En caso de superarlo... no sabemos lo que pasa, porque siempre nos mató!!!
Por eso le pedimos al lector invencible que nos cuente cómo es el final.

1985 MARTIN FIERRO V.2

TRUCO

PULSE UNA TECLA PARA COMENZAR EL JUEGO

NIVEL: 1 (PULSE F7)

TRUCO

Rating total: A
Creatividad: A
Documentación: B
Profundidad del juego: B
Valor en relación al precio: Se justifica
Mantiene el interés: Si
Computadora: Drean Commodore 64
Editor: ?

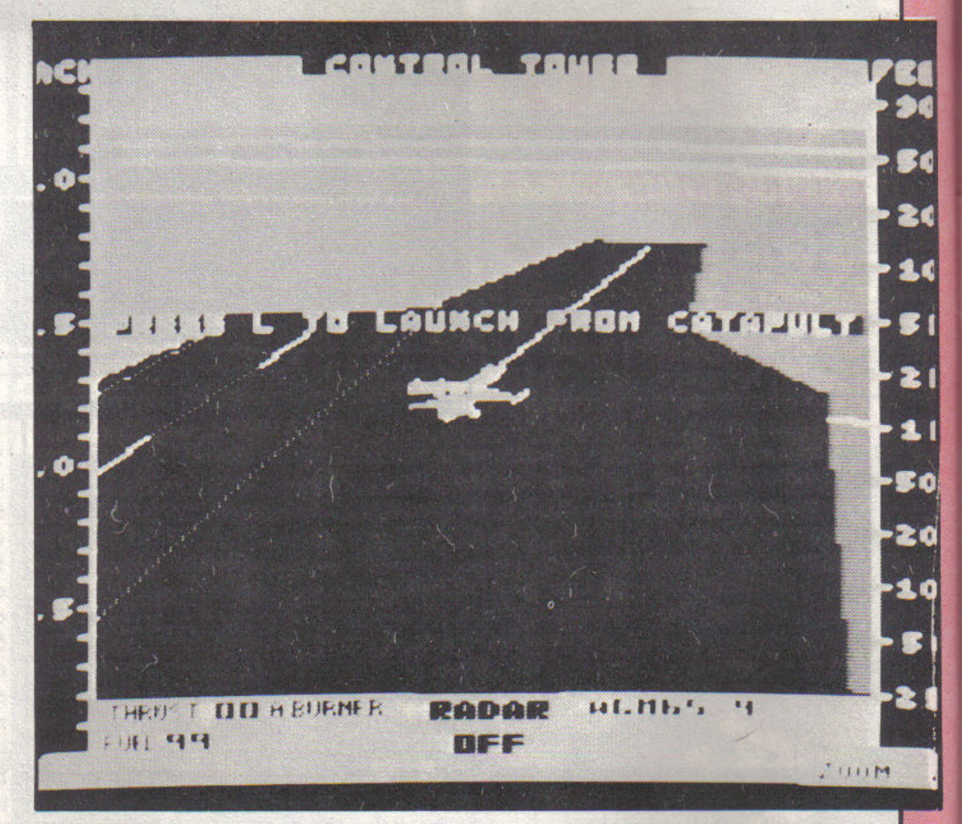
Aquellos que hayan recorrido INFOCOM'86 seguramente habrán escuchado una voz media rara que decía "truco", "falta envido", y demás comentarios referentes a este tradicional juego argentino.
Al acercarse, descubrieron con gran asombro que esa voz provenía de una C-64 que, cosa de locos, estaba jugando al truco con un oponente humano.
El juego comienza con la música del tango "El choclo" y agradeciendo la difusión del programa.
Lamentablemente la única información que aparece con respecto al autor del programa es nula, salvo un mensaje que asegura que "yo" es el autor.

Aquí se selecciona uno de los tres niveles de juego y se inicia la partida.
Las teclas que permiten ésto es F7 (el nivel) y oprimiendo cualquier tecla para empezar.
En la pantalla se imprimen dos casilleros denominados "Commodore 64" y "humano". Aquí se pone el score de cada uno de ellos.
Seguidamente se reparten las cartas siendo mano, la primera vez, el oponente.
Repartidas las cartas debemos jugar alguna de ellas o "cantar" algún desafío.
Lo primero se realiza a través de las teclas J, K y L, las cuales juegan la primera, segunda y tercera carta respectivamente.
Si, en cambio, queremos cantar truco, envido, etc., debemos oprimir las teclas correspondientes y luego return.
Por ejemplo, la "A" es flor, la "B" contraflor, la "C" con flor me achico, con "D" envido, con "E" real envido, con "F" falta envido, con "G" truco, con "H" quiero retruco, con "I" quiero vale cuatro, con "M" quiero, con "N" no quiero y con "O" me voy al mazo.
Jugada la carta o dada la orden, la computadora imprime "estoy pensando" y replica nuestra jugada.

Si dijimos "envido" ella puede contestar, siempre hablando, quiero o subir el tanto con la real o con falta envido.
Es decir que el juego se desarrolla como un verdadero partido de truco entre dos personas.
Si, por ejemplo, le decimos a la computadora que tenemos flor cuando en realidad no la tenemos, al principio nos dirá "usted gana" y nos incrementará el score.
Pero, al finalizar la partida, dará vuelta todas las cartas (incluyendo las de ella) y verá que no teníamos flor, por lo que nos dirá "tramposo", "momentito, momentito", "yo no soy tan gil como un humano" o "avivadas conmigo no".
Luego dirá "yo gano" y nos sacará de nuestro score los puntos respectivos.
El programa fue diseñado para permitir todos los ardides del juego y, como si esto fuera poco, la computadora puede mentir.
Cuando se genera un duelo del tipo "falta envido" el que tiene mayor cantidad de puntos para el tanto es el que gana, es decir no se diferencian las buenas de las malas.
Si quieren pasar un buen momento, escuchando como un "yankee" juega al truco, consigan este juego.

JET

Rating total: A
Creatividad: A
Documentación: B
Profundidad del juego: A
Valor en relación al precio: Se justifica
Mantiene el interés: Si
Computadora: Drean Commodore 64
Editor: SubLogic Corp.



Como en Flight Simulator II, SubLogic Corp. ha creado un nuevo juego que, en realidad, es un excelente simulador de combate.

Básicamente el objetivo del juego consiste en derribar aviones a través de un F-16 ó hundir buques enemigos utilizando un F-18.

Antes de comenzar, se nos pregunta si utilizaremos un televisor color o monocromático (blanco y negro).

Luego debemos seleccionar el modo de juego. Estos pueden ser Dog Fight, Target Strike, Free Flight, Demo Mode o Load Scenery Disk.

El primero de ellos (Dog Fight) consiste en derribar aviones enemigos a través de un F-16, despegando desde una base terrestre.

En el segundo (Target Strike) debemos destruir objetivos terrestres o marítimos.

El tercer y cuarto se refieren a vuelo libre (Free Fligth) y modo demostración (Demo Mode). El último modo nos permite cargar más escenarios de combate.

Una vez seleccionado el tipo de combate, debemos definir el nivel de dificultad del mismo. Estos pueden ser nivel 0, recomendado para practicantes, en donde no se choca contra el piso o contra el mar, pasando por el nivel 1 (fácil) hasta el nivel 9 (difícil).

Seleccionada la dificultad, tenemos que decir que avión usaremos. Estos pueder ser un F-16 (despegando desde una base terrestre) o un F-18 (despegando desde un portaaviones).

A continuación tenemos que armar el avión elegido con los distintos tipos de misiles.

Estos pueden ser AIM-9 (tipo aire-aire con radar propio y gran radio de acción), AIM-7 (tipo aire-aire, guiado por calor y pequeño radio de acción), AGM 65 (tipo aire-mar, busca por luz, autopropulsado); MK82 Bombas sin propulsión y, finalmente, una cinta de 500 proyectiles para una ametralladora de 200 mm.

No debemos olvidar que a mayor cantidad de armas almacenadas en el avión, mayor será el peso de éste lo cual se notará cuando tengamos que despegar.

En caso de haber seleccionado un F-18, y luego de armarlo, tenemos que despegar para atacar los buques enemigos.

A partir de aquí entran en juego los distintos comandos que dispone el piloto para el manejo de su jet. Estos permiten visualizar el escenario de combate atrás, hacia la derecha, hacia la izquierda y hacia arriba (por supuesto, también para adelante).

Para despegar del portaaviones debemos acelerar, a través de "+", los motores al máximo. Luego, oprimir la tecla "L" para que se nos "catapulte".

Veremos cómo el indicador de velocidad, a la derecha de la pantalla, comienza a aumentar y como vamos ganando altura.

Si seguimos ascendiendo, realizaremos un "loop", el cual dañaría al piloto por los cambios bruscos de aceleración.

Si logramos estabilizar el avión, debemos conmutar el radar de combate y buscar los buques enemigos.

Si nuestro avión es un F-16 tenemos que, antes de iniciar el carretaje, sacarlos del hangar.

Luego debemos darle propulsión para tomar altura, y conectar el radar de combate. La mira se activa a través de la tecla "R".

Para disparar un misil o una bomba, el contorno de la mira debe estar totalmente en marrón. Esto indicará que el avión enemigo está al alcance de los misiles.

Los gráficos o, mejor dicho, los paisajes que se observan desde la cabina fueron diseñados para simular montañas, ríos y pistas de aterrizaje. Todas en tres dimensiones.

Para aquellos que gusten de juegos con acción contínua, les recomendamos este excelente "juego".

SKYFOX



Rating Total: B
Creatividad: A
Documentación: B
Profundidad de juego: A
Valor en relación al precio: Se justifica
Mantiene el interés: Si
Computadora: Drean Commodore 64
Editor: Electronics Art

SKYFOX es otro simulador de combate cuyo objetivo es derribar aviones y destruir tanques enemigos.
Para ello, el piloto dispone de una consola totalmente equipada con indicador de altitud, indicador de velocidad, brújula magnética, indicadores de coordenada de la ubicación del enemigo, indicador de nuestra posición dentro del escenario de combate y, finalmente, la computadora de abordo.
Pero antes de seguir comentando los objetivos del juego, tenemos que mencionar los gráficos o, mejor dicho, los escenarios de combate.
Desde ya éstos dan un efecto tridimensional y, a diferencia de JET, la velocidad de movimiento es muy alta. Cada uno de ellas debe ser utilizado de acuerdo a la situación de combate, las cuales pueden ser 15.
Los virajes del avión, de un lado hacia el otro, se realizan con tal rapidez que más que una simulación es, prácticamente, una situación real.
En lo que respecta al armamento del SKYFOX, el avión es capaz de disparar láser, misiles guiados por radar y misiles guiados por calor.
Ellas van desde una misión de entrenamiento hasta una invasión total. Además, por cada uno de ellos, existen cinco niveles los cuales van desde "cadete" hasta el "as de la base".

Omitimos mencionar el radar que dispone el SKYFOX el cual nos permite saber la posición de nuestro avión y la del enemigo.
Cuando ya lo hemos localizado, éste puede ser conmutado a mira, que contribuirá a que el láser o los misiles den en el blanco.
El movimiento del avión se hace a través del joystick. Además se utiliza el teclado para poder utilizar todas las posibilidades del SKYFOX.
Una de ellas es la computadora de abordo que se acciona a través de la tecla "C". Por ella se conmuta el paisaje actual a las distintas opciones que posee.
Por ejemplo, podemos ver el mapa total del escenario de lucha, desarmar o armar los misiles, cambiar la ruta de vuelo y, si no nos acordamos de todos los comandos que permite, podemos oprimir la tecla "H" la cual nos mostrará todos ellos.

Todo ésto se hace durante el vuelo. En caso de que nuestro avión reciba demasiados impactos del enemigo, la computadora de abordo se destruirá.

La velocidad crucero del avión puede ser seleccionada a través de las teclas 0-9. cada una de ellas aumenta la velocidad proporcionalmente, permitiéndonos "flotar" en el aire hasta desarrollar velocidades supersónicas.
A través de la barra espaciadora aumentamos la velocidad de nuestro avión.
Otra tecla de gran importancia, que se puede oprimir en cualquier momento, es la "A". Esta busca automáticamente los objetivos y devuelve el control al piloto.
Todos los efectos reales son perfectamente simulados por SKYFOX. Por ejemplo, cuando vamos ascendiendo, las nubes pasan por nuestro costado apareciendo, a los 1000 pies, un banco de nubes.
Las instrucciones de los objetivos del juego vienen acompañadas con el mismo juego. Es decir que podemos cargarlas en la computadora y estudiar todo lo deseado.

Como habrán notado, SKYFOX es más que un simple juego. Seguimos insistiendo en la calidad y velocidad de los gráficos. Lo recomendamos.

CORREO-CONSULTAS

Ingreso por teclado

Soy un flamante usuario de una C-128 y, como estudiante de ingeniería, me interesa fundamentalmente la aplicación matemática de mi computadora y como en la revista número 3 había un programa que resolvía integrales y raíces de ecuaciones me decidí a investigarlo.
Dejando de lado las sentencias POKE, que son independientes para cada computadora, y, además, ninguna de las incluídas hacen que se detenga para ingresar la función a integrar o cuyas raíces se quieran extraer.
¿Hay alguna manera de ingresar esa función, o generalmente alguna sentencia en un programa y que luego éste se pueda hacer correr a partir de ese punto sin perder los datos ya ingresados?
Además, quisiera saber si existe en el país bibliografía para la C-128 y en caso de no haberla (aparentemente lo único que hay son libros que se remiten a traducir el manual del usuario) se puede usar para el modo C-64 la bibliografía del C-64.

Sin otro particular y agradeciendo el haber encontrado una revista para usuarios de Commodore que no se dedique sólo a mostrar juegos, saludo a ustedes muy atentamente.

Daniel Santiago Vallespir
Camacuá 144
Capital Federal

En el número 7 hemos publicado un método para poder ingresar funciones por teclado.
Con el mismo criterio, y conociendo muy bien cómo se almacenan los programas en memoria, se puede ingresar una sentencia por programa y reanudar la ejecución del mismo.

Por ello te pedimos que leas las notas sobre "Evaluación de funciones" y "Almacenamiento de los programas".
Con respecto a la bibliografía que mencionas, lo único que conocemos es un libro de la serie Data Becker el cual habla muy por encima de los aspectos más importantes de la C-128.
Tenemos entendido que, de la misma serie, llegarán libros con información más específica.
Si utilizas la C-128 en modo 64, la bibliografía que existe para esta última es utilizable.

Continuamos con esta sección para que los lectores planteen sus consultas y sugerencias. Para eso deben escribir a Revista para usuarios de Drean Commodore, Paraná 720, 5to. Piso, (1017) Cap. Federal.

Telegrafía y juegos

Soy "usuario" de una C-64 desde febrero del corriente año y realmente la única utilidad que le doy es pasarle juegos a mi hijo de 5 años. La falta de tiempo me impide iniciar algún curso de programación, razón por la que soy lector de vuestra revista desde el número 1. Veo con alegría que en el número 5 comienzan una sección para los que recién se inician; he tratado de copiar programas de revistas, pero no logre más que un ?Sintax error in... pero cuando finalice mis estudios Universitarios (en junio Dios mediante) me dedicaré de lleno a este apasionante tema.

Mis preguntas:
1) ¿Hay alguna posibilidad de que los juegos pasen de nivel sin el uso del Joystick a los efectos de conocer el juego en toda su magnitud?
2) Soy Radioaficionado, poseo un equipo Yaesu FT707, ¿hay alguna forma de conectar el C-64 para recibir y trasmitir en telegrafía? Sé que colegas LU utilizan la TI 99.
Muchas gracias y sigan adelante.

Esteban Eduardo Cejas
Azara 1006
Banfield

Hay juegos que disponen de un modo que se llama Demo, es decir demostración. En él se muestra todo el desarrollo del mismo.
En caso de que no tengan esta opción, no se puede saber cómo es el juego en su totalidad sin participar en el mismo.
Con respecto a la segunda pregunta la respuesta es sí. El único inconveniente es que se deben tener conocimientos sobre el Port del usuario de la C-64, tema que será incluído en futuras notas.

Sugerencias

Ante todo debo felicitarlos por la publicación de esta revista, es muy interesante.
Les escribo con el fin de hacerles llegar una sugerencia. Mi nombre es Gustavo Haissiner, tengo 14 años y me interesa mucho esta revista porque soy usuario de una Drean Commodore 16. Me gustaría saber si podrían publicar material sobre mi computadora que no esté en su manual ya que Gráficos de Alta Resolución, los comandos HELP, DIRECTORY, etc. están dentro de él.

Gustavo Haissinger
Billinghurst 2407
Capital